VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Automatický převod MS Access aplikací do prostředí .NET

Automatic Conversion of MS Access applications to .NET Framework

2017 Marek Fajstl

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání diplomové práce

Student: **Bc. Marek Fajstl**

Studijní program: N2647 Informační a komunikační technologie

Studijní obor: 2612T025 Informatika a výpočetní technika

Téma: Automatický převod MS Access aplikací do prostředí .NET
Automatic Conversion of MS Access applications to .NET Framework

Jazyk vypracování: čeština

Zásady pro vypracování:

Cílem práce je návrh a implementace softwarového rámce pro automatický převod aplikací vyvíjených v prostředí Microsoft Access s napojením na SQL Server do prostředí Microsoft .NET Framework. Prototypová implementace bude obsahovat převod vybraných objektů.

Práce bude splňovat následující body:
1. Rozbor objektů v aplikaci pro MS Access a .NET.
2. Rešerše stávajících řešení.
3. Analýza chování uživatelských formulářů v MS Access i .NET z hlediska komunikace se SŘBD.
4. Rozbor možností programového přístupu k aplikaci (zjišťování seznamu formulářů, tiskových sestav, maker, jejich komponent, vlastností, atd.).
5. Návrh a implementace prototypové aplikace pro automatický převod.
6. Demonstrace implementace na zvolených MS Access aplikacích, shrnutí výhod a nevýhod a porovnání s existujícími nástroji.

Seznam doporučené odborné literatury:

Podle pokynů vedoucího diplomové práce.

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **Ing. Petr Lukáš**

Datum zadání: 01.09.2015

Datum odevzdání: 28.04.2017

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 28. dubna 2017

Marek Fogell

.....

Souhlasím se zveřejněním této diplomové práce dle požadavků čl. 26, odst. 9 Studijního a zkušebního řádu pro studium v magisterských programech VŠB-TU Ostrava.

V Ostravě 28. dubna 2017

Marek Fajfík

.....

Rád bych poděkoval vedoucímu práce Ing. Petru Lukášovi za pomoc, trpělivost, věcné připomínky a vstřícnost při konzultacích.

Abstrakt

Cílem této diplomové práce je navrhnout a implementovat konverzní nástroj pro automatický převod Microsoft Access aplikací do prostředí Microsoft .NET Framework. Na začátku práce je uveden rozbor objektů Microsoft Access aplikací a jejich anologie v prostředí .NET. Dále je zde popsána komunikace MS Access formulářů a formulářů v .NET Framework se SŘBD. V práci jsou také popsány již existující nástroje pro zmíněnou konverzi a postup pro konverzi tiskových sestav. Poslední část je zaměřena na popis vlastního konverzního nástroje a celkové zhodnocení práce.

Klíčová slova: Konverze Microsoft Access aplikací, Microsoft Access, .NET Framework, Windows Forms, SQL Server Reporting Services

Abstract

The main aim of the thesis is to design and implement conversion tool for automatic transform Microsoft Access application to .NET Framework. At the beginning of the thesis there is analysis of Microsoft Access objects and their analogy in .NET. The next part describes communication of MS Access forms and forms in .NET framework with DBMS. This thesis also describes existing conversion tools deals with mentioned transform and process for conversion reports. The last part deals with a description of my own variant of conversion tool and overall evaluation of the work.

Key Words: Conversion Microsoft Access application, Microsoft Access, .NET Framework, Windows Forms, SQL Server Reporting Services

Obsah

Seznam použitých zkratek a symbolů

CRUD	–	Create, Read, Update, Delete
DBMS	–	Database Management System
DLL	–	Dynamic-link library
MS	–	Microsoft
ODBC	–	Open Database Connectivity
OLE DB	–	Object Linking and Embedding, Database
ORM	–	Object-relational mapping
RDL	–	Report Definition Language
SQL	–	Structure Query Language
SŘBD	–	Systém Řízení Báze Dat
SSRS	–	SQL Server Reporting Services
URL	–	Uniform Resource Locator

Seznam obrázků

Seznam tabulek

Seznam výpisů zdrojového kódu

1 Úvod

Prakticky v každém podniku nebo organizaci se v současnosti používá nějaký informační systém. S nárůstem množství dat se tyto systémy stávají rozsáhlejší a složitější. Pro mnohé podniky nebo organizace není zpočátku zapotřebí uchovávat velké množství informací a počítá se například pouze s několika tabulkami přístupnými přes odpovídající formuláře. Pro takové společnosti je vhodnou volbou například nástroj od firmy Microsoft a to Microsoft Access. Vývoj v tomto produktu je rychlý, snadný a uživatelsky přívětivý.

Takové řešení se mnohdy osvědčilo a informační systém začal v průběhu let nabírat na velikosti a složitosti. K systému se začalo připojovat více uživatelů, bylo potřeba je ověřovat a spravovat uživatelské role. Východiskem ze situace, kdy Microsoft Access databáze již nedostačuje potřebám společnosti a přitom tato společnost nechce upustit od svého osvědčeného software, je převedení dat na MS SQL Server. Pro tuto migraci existuje mnoho nástrojů, a pokud je Microsoft Access databáze správně koncipovaná, může informační systém sloužit o dalších pár let déle.

Problém nastává v roce 2013 s příchodem nového kancelářského balíku MS Office 2013, jehož je Microsoft Access součástí. V této verzi Microsoft přestal podporovat nativní přístup k Microsoft SQL Serveru. Jednou z možností pro vyřešení tohoto problému je použít pro přístup do databáze ODBC, což s sebou nese řadu problémů a zároveň propustnost takového řešení není nejlepší. Druhá možnost spočívá v upuštění od dosavadního řešení, následného zakoupení univerzálního informačního systému a jeho přizpůsobení potřebám podniku. Třetí možností je pokusit se tuto Microsoft Access aplikaci převést na jinou platformu a zachovat tak nastavené podnikové procesy.

Cílem této práce je navrhnout takový konverzní nástroj, který dokáže převést stávající Microsoft Access aplikaci do novější a do budoucna rozšiřitelnější podoby. Rámcem pro tvorbu takové aplikace byl zvolen .NET Framework, který je svou rozsáhlostí knihoven a bezpečností vývoje ideální pro další rozvoj dané aplikace.

První část práce je věnována popisu objektů Microsoft Access aplikace, jejich komunikaci se SŘBD a také jejich ekvivalentu v aplikačním rámci .NET. Jsou zde popsány i existující konverzní nástroje.

V další části jsou rozebrány možnosti programového přístupu k aplikaci Microsoft Access. Je zde uveden postup získání objektů z původní Access aplikace pro následné využití v konverzním nástroji. Dále je zde uveden postup převodu tiskových sestav do podoby pro výslednou aplikaci v aplikačním rámci .NET.

V poslední části práce je popsán vytvořený konverzní nástroj, použité knihovny a podrobný postup pro konverzi pomocí tohoto nástroje. Je zde uveden i příklad konverze jednoduché Microsoft Access aplikace a na závěr je provedeno porovnání s již existujícími nástroji.

2 Microsoft Access

Microsoft Access (dále jen MS Access) je systém pro správu relačních databází od společnosti Microsoft, který kombinuje relační databázový stroj Microsoft Jet Database Engine s grafickým uživatelským rozhraním. První verze tohoto nástroje byla vydána v roce 1992. Od verze MS Access 95 se aplikace stala součástí balíku Microsoft Office. [2]

MS Access umožňuje přistupovat k datům z databází Jet Database Engine, Microsoft SQL Server, Oracle či ke kterékoliv další databázi přes rozhraní ODBC (včetně MySQL a PostgeSQL). Softwaroví vývojáři a datoví architekti jej mohou použít k vývoji aplikačního software a pokročilí uživatelé k tvorbě jednoduchých aplikací. MS Access podporuje některé objektově orientované techniky, ale nepostačuje jako plně objektový vývojový nástroj. [4]

V MS Access aplikacích existují dva typy souborů MDB a ADP. V souborech typu ADP (od verze MS Access 2000) je použita architektura klient-server databáze s databází SQL Server na místě databáze Jet. V takovém případě pak aplikace podporuje tvorbu téměř všech objektů použitého serveru. Formuláře, sestavy, makra a moduly jsou tak uloženy v ADP souborech a ostatní objekty v serverové databázi. Od verze MS Access 2013 se od této podpory upustilo a bez složitého zásahu do návrhu aplikace nelze tyto aplikace v novějších verzích používat. MDB soubory obsahují všechny části jako ADP, ale navíc jsou v nich uložena samotná data, pohledy, funkce, uložené procedury a nastavení zabezpečení databáze. Tyto soubory pak mohou sloužit i jako databáze, která je přístupná přes rozhraní ODBC jiným aplikacím. Typicky lze tabulky obsažené v těchto souborech importovat do aplikace Microsoft Excel. [3], [1]

MS Access aplikace se skládá z množiny objektů následujících typů: *formuláře*, *ovládací prvky*, *uživatelské menu*, *sestavy*, *makra* a *moduly*. V následujících podkapitolách jsou jednotlivé objekty podrobně rozebrány.

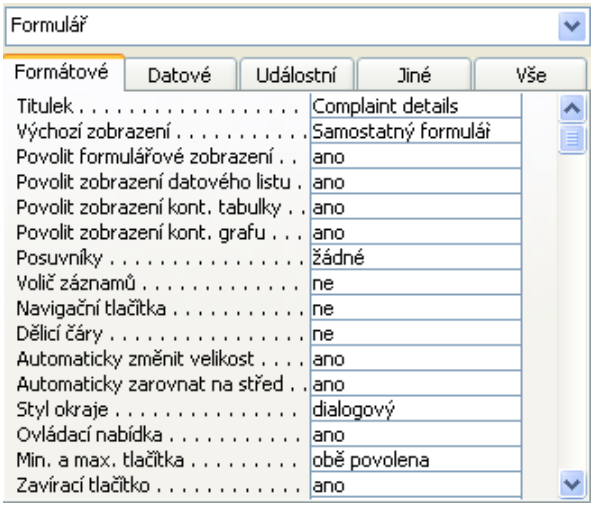
2.1 Formuláře

Formuláře v MS Access aplikacích jsou určeny pro práci s daty a tvoří uživatelské rozhraní aplikace. Jsou založené na údajích z tabulek a dotazů. Mohou též obsahovat vlastní výpočty. Existují i formuláře bez údajů, které se používají hlavně k vytvoření navigačního systému v databázi, aby koncový uživatel mohl s aplikací efektivně pracovat.

Formuláře lze doplnit různými ovládacími prvky, jako jsou *tlačítka*, *rozevírací seznamy* a další (jednotlivé prvky jsou podrobněji rozebrány v kapitole 2.2). Pomocí maker nebo modulů lze těmto ovládacím prvkům přiřadit funkční logiku. Znamená to, že po klepnutí na tlačítko se například otevře jiný formulář nebo sestava.

Formulář je možné také vytisknout, i když k tomuto účelu jsou primárně určeny tiskové sestavy. Při zobrazení datového listu (viz kapitola 2.1.2) je tisk podobný jako při tisku tabulky v tabulkovém procesoru (např. Microsoft Excel). V jiných zobrazeních jako je samostatný formulář (viz kapitola 2.1.1), by se tisklo i případné barevné pozadí formuláře a to není vždy vhodné.

Všechny formuláře mají i své vlastnosti, které lze nastavovat buď pomocí panelu v návrhovém zobrazení, makrem nebo kódem VBA. Tyto vlastnosti jsou rozděleny na formátové, datové, událostní a jiné. Na obrázku 1 je zobrazen panel pro nastavení vlastností formuláře. [2]



Obrázek 1: Vlastnosti formuláře

Název	**Popis**
Titulek	Text zobrazený v titulku.
Výchozí zobrazení	Tato vlastnost slouží pro nastavení zobrazení dat na formuláři. Nejdůležitější a nejpoužívanější hodnoty této vlastnosti jsou *Samostatný formulář* a *Datový list*. *Samostatný formulář* je klasický formulář popsaný v kapitole 2.1.1 a *Datový list* se používá pro zobrazení tabulky (viz kapitola2.1.2).
Navigační tlačítka	Povolení či zákaz zobrazení navigačních tlačítek ve formuláři.
Volič záznamů	Povolení či zákaz zobrazení voliče záznamů ve formuláři.
Zdroj formuláře	Název tabulky nebo dotazu, který je k formuláři připojen jako zdroj dat. Může zde být uveden také SQL dotaz.
Filtr	Výraz pro filtrování dat.
Řadit podle	Kritérium řazení záznamů.
Povolit úpravy	Povolení úprav zdrojových dat ve formuláři. Tuto vlastnost můžeme zakázat např. určité skupině uživatelů.
Povolit odstranění	Povolení či zákaz odstranění záznamů z tabulky.
Povolit přidávání	Povolení či zákaz přidávání nových záznamů do tabulky.
Překrývané okno	Pokud je vlastnost nastavena na *Ano*, zůstává formulář vždy navrchu pracovní plochy MS Access, i když není zrovna aktivní. Překryvný formulář může být modální i nemodální.

Pokračování tabulky na další stránce

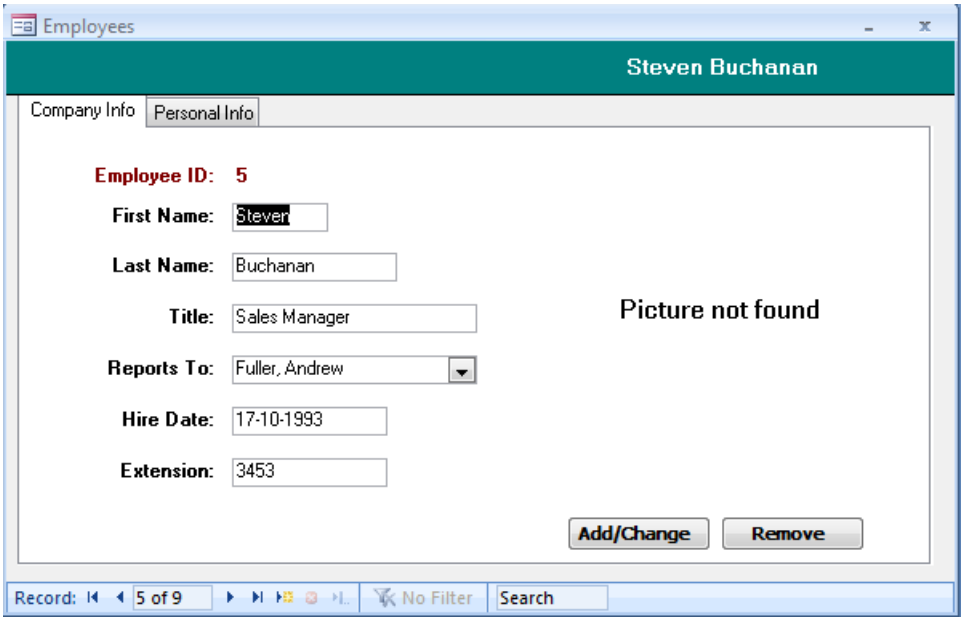
Název	Popis
Modální okno	Vlastnost *Modální okno* určuje, zda se formulář otevře jako modální. To znamená, že před přechodem z formuláře do jiného okna databáze musí být celá aktivace formuláře ukončena.

Tabulka 1: Seznam důležitých vlastností formuláře

2.1.1 Samostatný formulář

Samostatný formulář se používá nejčastěji na prohlížení, přidávání a úpravu údajů z jedné tabulky. Formulář také může obsahovat i podformulář, čímž lze realizovat rozhraní pro úpravu obsahu tabulky ve vztahu 1:N. Záznamy jsou v jednoduchém formuláři zobrazeny po jednom na samostatných stránkách.

Samostatný formulář se skládá z několika součástí. Příklad takového formuláře je vidět na obrázku 2. V horní části formuláře se nachází hlavička, která obsahuje ikonu a titulek formuláře vytvořený z názvu zdrojového objektu nebo názvu samotného formuláře. Údaje jsou zobrazeny v části *Tělo formuláře* v ovládacích prvcích vytvořených z polí tabulky nebo dotazu. Názvy polí na formuláři jsou použity ze zdrojové tabulky nebo dotazu. V dolní části jsou navigační tlačítka, pomocí kterých se lze přesouvat mezi jednotlivými záznamy a tlačítkem s hvězdičkou lze přidat nový záznam. Ve spodní části je i vyhledávací pole, které zobrazuje shodné záznamy postupně, jakmile se vypisuje hledaný text. [1]

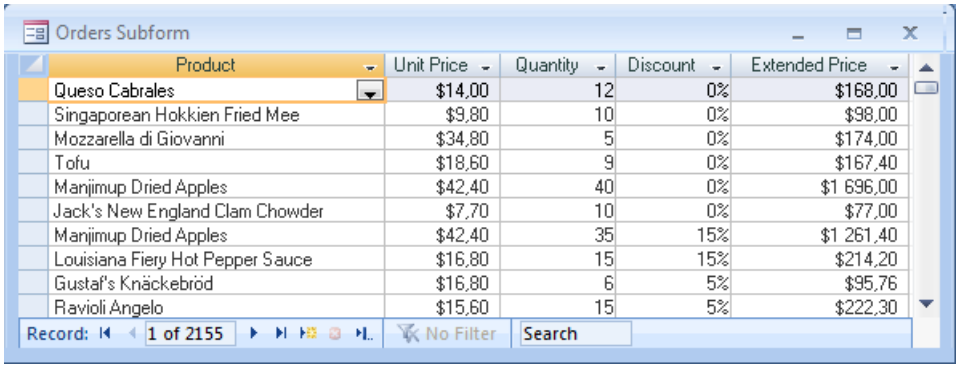


Obrázek 2: Samostatný formulář

2.1.2 Datový list

Datový list zobrazuje údaje tabulkovou formou, kde se každý záznam nachází na samostatném řádku. Záznamy lze pomocí datových listů vkládat, upravovat a odstraňovat. Kromě toho lze provádět i výpočty podobně jako v databázových dotazech. Taktéž lze nastavit, které údaje budou viditelné nebo budou jen ke čtení a zamknuté.

Takový typ formuláře zobrazuje jednu kontextovou kartu. Lze měnit motivy, přidávat pole, měnit barvu pozadí a řádku a taktéž lze využít podmíněné formátování. [1]



Obrázek 3: Datový list

2.2 Ovládací prvky

2.2.1 Popisek

Popisek je ovládací prvek zobrazující text. Na rozdíl od *textového pole* nelze tento text ve formulářovém zobrazení editovat. Všechny následující vlastnosti platí i pro ostatní ovládací prvky. [2]

Název	**Popis**
Titulek	Text, který ovládací prvek zobrazuje.
Zobrazit	Povolí či zakáže zobrazení ovládacího prvku na formuláři.
Vlevo	Hodnota vyjadřující vzdálenost (v centimetrech) levého okraje ovládacího prvku od horní části formuláře, na kterém je ovládací prvek umístěn
Nahoře	Hodnota vyjadřující vzdálenost (v centimetrech) horního okraje ovládacího prvku od horní části formuláře, na kterém je ovládací prvek umístěn.
Šířka	Šířka ovládacího prvku v centimetrech.
Výška	Výška ovládacího prvku v centimetrech.

Tabulka 2: Seznam důležitých vlastností *Popisku*

Pokud je nastavená vlastnost *Hypertextový odkaz*, popisek bude plnit funkci tlačítkového odkazu a po kliknutí otevře v prohlížeči příslušnou webovou stránku, nebo pokud je textem emailová adresa, otevře emailového klienta pro napsání nové zprávy na zadanou adresu.

2.2.2 Textové pole

Textové pole je nejčastěji používaný ovládací prvek. Zobrazuje většinu údajů od textových přes číselné, měnové, datové až po vypočítané. *Textové pole* se skládá z buňky pro zobrazení textu a volitelně pak z vlastního přidruženého popisku. [2]

Název	**Popis**
Posuvníky	Je možno zvolit buď *žádné* nebo *svislé*. Podle toho bude či nebude v poli možnost vertikálního posuvu pomocí posuvníku.
Zdroj ovládacího prvku	Zde bývá uveden zdroj dat. Vlastnost může obsahovat i výraz nebo příkaz SQL.
Zpřístupnit	Hodnota *Ano* umožňuje uživateli data v tomto poli (a tím i v podkladové tabulce) měnit. Pokud je vlastnost nastavena na hodnotu *Ne*, je textové pole zašedlé a neaktivní.
Název	Název, pomocí kterého se na pole odkazujeme ve výrazech, příkazech SQL, makrech a programových modulech.
Text na stavovém řádku	Kontextová nápověda, která se zobrazí ve stavovém řádku pracovní plochy MS Access, pokud je ovládací prvek zaměřen.
Před aktualizací	Tato událost je vyvolána před aktualizací dat v poli. Pokud se změní text zapsaný v poli, promítnou se tyto změny do podkladové tabulky dat teprve ve chvíli aktualizace celého pole. Aktualizací lze dosáhnout například (a nejčastěji) přenesením zaměření na jiný ovládací prvek formuláře (klepnutím myší na jiný prvek nebo stisknutím klávesy *ENTER* nebo *TAB*). Jako hodnotu této vlastnosti lze definovat odkaz na proceduru v modulu formuláře, která zkontroluje platnost dat.
Při změně	Událost při změně nastane pokaždé, když uživatel provede změnu přímo v poli, tj. ihned při zápisu nebo smazání každého jednotlivého znaku. Následuje velké množství událostí spouštěných např. při přesunutí kurzoru myši nad objektem, Při klepnutí myši atd.

Tabulka 3: Seznam důležitých vlastností prvku *Popisek*

2.2.3 Zaškrtávací políčko

Na formulářích se nacházejí i *zaškrtávací políčka*. Pokud se zobrazují pole z tabulky, tak pro každý záznam mohou mít jiný stav - zaškrtnuté/nezaškrtnuté. Jedná se vlastně o ovládací prvek pro zobrazení nebo úpravu hodnoty datového typu *bool*.

Název	**Popis**
Trojnásobný stav	Políčko může být zaškrtnuté, nezaškrtnuté a bez stavu (hodnota NULL)

Tabulka 4: Seznam důležitých vlastností prvku *Zaškrtávací políčko*

2.2.4 Přepínač

Přepínač plní v MS Access stejnou funkci jako *zaškrtávací políčko* a dokonce mají i stejné vlastnosti. Jediným rozdílem je, že přepínač se primárně používá pro výběr jedné z mnoha navzájem vylučujících se možností. Pro takovou funkčnost je použit ovládací prvek *skupina kontrolních prvků*, do které musí být přepínače umístěny. Při zaškrtnutí jednoho přepínače, se pak odškrtnou všechny ostatní. Nicméně tohle lze provést i s prvky typu *zaškrtávací políčko*.

2.2.5 Pole se seznamem

Pole se seznamem, neboli rozbalovací seznam, je velice podobný ovládacímu prvku *textové pole*. V pravé části má ovšem šipku, která slouží k rozbalení seznamu hodnot, ze kterých uživatel může vybrat hodnotu pole.

Název	**Popis**
Počet řádků seznamu	Číselná hodnota udává počet řádků rozbaleného seznamu. Pokud má seznam více položek, je zobrazen se seznamem i posuvník.
Šířky sloupců	Určuje šířky jednotlivých sloupců, oddělené středníky. Pokud je zadaná hodnota 0, pole bude skryto. V poli se seznamem se první viditelný sloupec zobrazí v textové části ovládacího prvku. Datový typ libovolné hodnoty napsané v poli se seznamem musí být stejný nebo slučitelný s datovým typem prvního viditelného sloupce.
Typ zdroje řádků	Možnost výběru z tohoto výčtu možností: *tabulka nebo dotaz*, *seznam hodnot* a *seznam polí*

Pokračování tabulky na další stránce

Název	Popis
Zdroj řádků	V případě, že *Typ zdroje řádků* je nastaven na *tabulka nebo dotaz* nebo *seznam polí*, je nutno do tohoto parametru vložit SQL dotaz. V případě *seznam hodnot*, je nutno zadat přímo hodnoty seznamu oddělené středníkem (např. `CZK;USD;EUR`).
Vázaný sloupec	Tato vlastnost určuje, který sloupec je vázaný k podkladovému poli, a zadává se zde pořadí tohoto prvku v SQL dotazu. Například pokud máme dotaz `select id,jmeno from Zamestnanci`, a id je vázaný sloupec, bude hodnota tohoto parametru 1.
Omezit na seznam	Určuje, zda pole se seznamem přijímá libovolný zadaný text nebo jen text odpovídající jedné z hodnot v seznamu.
Automaticky doplnit	Určuje, zda program MS Access automaticky vyhledá a doplní hodnotu odpovídající znakům zadaným v poli se seznamem v průběhu psaní.

Tabulka 5: Seznam důležitých vlastností prvku *Pole se seznamem*

2.2.6 Příkazové tlačítko

Jedná se o klasické tlačítko např. „OK“ nebo „Storno“ v nejběžnějších dialozích.

Název	Popis
Zpřístupnit	Pokud bude hodnota této vlastnosti nastavena na *Ne*, bude tlačítko šedé a tudíž nepřístupné.
Při klepnutí	Tato událost nastane při klepnutí levým tlačítkem myši na tlačítko.

Tabulka 6: Seznam důležitých vlastností prvku *Příkazové tlačítko*

2.2.7 Karta

V aplikaci MS Access lze také vytvořit formulář s několika stránkami. K tomuto slouží ovládací prvek *Karta*, kde lze nastavit libovolný počet navzájem se překrývajících karet. *Karta* neobsahuje vlastnosti důležité pro převod.

2.2.8 Obdélník

Pro zpřehlednění prvků na formuláři lze dále použít ovládací prvek obdélník, který nemá žádnou funkcionalitu, a pouze vykreslí obdélník kolem určité oblasti prvků. *Obdélník* neobsahuje vlastnosti důležité pro převod.

2.3 Uživatelské menu

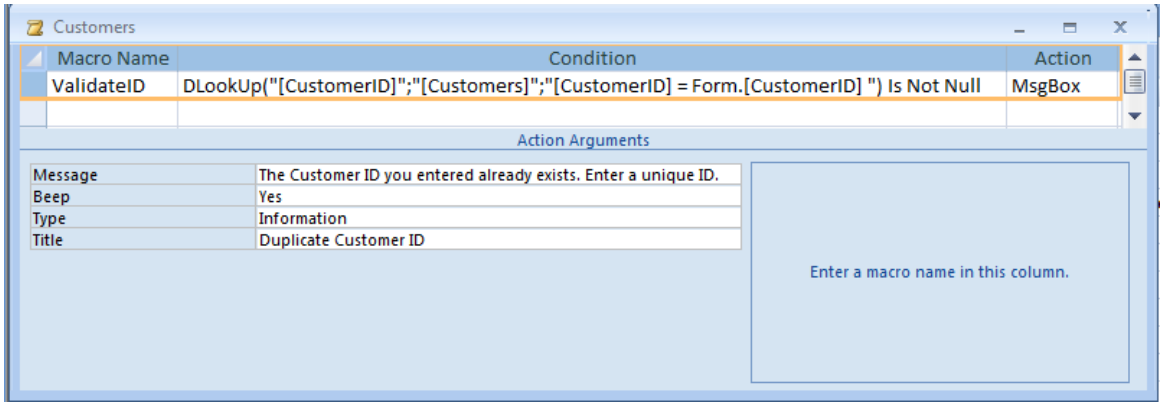
V dřívějších verzích MS Access se pro definování uživatelské nabídkové lišty používala posloupnost maker. Bylo zapotřebí jedno makro, které se skládalo z několika akcí *PřidatNabídku*, a které definovalo základní nabídkovou lištu. Poté byla zapotřebí další makra, pro každou roletovou nabídku či podnabídku jedno makro. Ve verzích 2000 a 2003 se tato nabídka dala graficky navolit přímo v aplikaci MS Access. Od verze 2007 se od tohoto řešení upustilo a Microsoft přišel s novou filozofii inovativního ovládání uživatelského rozhraní *Ribbon* s pásem nástrojů. Jedná se o poměrně široký panel nástrojů, umístěný v horní části okna. Ikony v něm se přizpůsobují práci. Takovým pásem jsou vybavené nejdůležitější aplikace kancelářského balíku MS Office. [1] Pro převod MS Access aplikací do prostředí .NET, budeme uvažovat pouze standardní MS Access menu.

2.4 Sestavy

Sestavy se nejčastěji používají na tiskové výstupy údajů z tabulek nebo dotazů. Využit je lze samozřejmě i k prohlížení údajů na obrazovce. Narozdíl od formulářů sestavy neslouží k úpravě dat. Stejně jako formuláře, i sestavy mohou obsahovat vložené sestavy a zobrazovat tak údaje z propojených tabulek. [1]

2.5 Makra

Makro je objekt, který je strukturovanou definicí jedné nebo více akcí, které chceme, aby MS Access vykonal jako odezvu na nějakou definovanou událost. Lze například navrhnout makro, jež bude jako odezvu na vybrání nějaké položky v hlavním formuláři otevírat druhý formulář. Dále lze mít makra, která budou při změně hodnot v polích ověřovat jejich obsah. Do maker lze zařadit jednoduché podmínky, kterými se specifikuje, kdy má být jedna či více akcí provedena nebo přeskočena. Pomocí maker lze otevírat a provádět dotazy, otevírat tabulky nebo tisknout či prohlížet sestavy. Z maker lze také spouštět další makra nebo procedury VBA. K vytváření maker slouží nástroj *Tvůrce maker*, který je zobrazen na obrázku 4. [2]



Obrázek 4: Makro, které provádí kontrolu zda nebylo vepsáno již použité ID

2.6 Moduly

Modul je objekt obsahující uživatelské procedury, které jsou napsány v jazyce Visual Basic for Applications (VBA). Jazyk VBA je používán ve všech aplikacích balíčku Microsoft Office. Tento programovací jazyk umožňuje vytváření uživatelsky definovaných funkcí, automatizaci procesů, přístup k Windows API a ostatní nízkoúrovňové funkce prostřednictvím dynamicky linkovaných knihoven (DLL). Program v jazyce VBA může být použit pro kontrolu mnoha aspektů hostující aplikace, včetně manipulace s funkcemi uživatelského rozhraní. Dovoluje také pracovat s vlastními uživatelskými formuláři a dialogy, tj. dokáže zjišťovat jejich vlastnosti a komponenty, ze kterých se skládají. Jak již název tohoto jazyka napovídá, VBA je velmi úzce příbuzné s Visual Basic a používá stejnou běhovou knihovnu Visual Basic Runtime Library, avšak program napsaný v tomto jazyce dokáže běžet pouze spolu s hostující aplikací, nikoliv jako samostatný program. Kód, který je napsán ve VBA, je přeložen do proprietárního prováděcího kódu Microsoft P-code (z anglického packed code, což znamená zabalený kód). Tento kód hostující aplikace (např. Microsoft Access nebo Microsoft Excel) ukládá jako samostatný datový proud v samotném dokumentu (například .ADP). Kód je poté spuštěn ve virtuálním stroji, který zajišťuje hostující aplikace. [10]

Moduly nabízejí oproti makrům více chráněný tok akcí a umožňují zachycovat chyby, což s makry provádět nelze. Moduly mohou být samostatné objekty obsahující funkce, které lze zavolat z libovolného místa v aplikaci, pak se jedná o tzv. Globální moduly. Moduly dále mohou být přímo přidruženy k nějakému formuláři nebo sestavě a odpovídat pouze na události tohoto přidruženého formuláře či sestavy. Dále existují speciální typy modulů tzv. třídy. MS Access tedy nabízí i základní podporu objektově orientovaného programování, ale nepostačuje jako plně objektový vývojový nástroj. [2]

3 .NET Framework

.NET Framework je aplikační rámec skládající se z několika elementů [6]:

- společné běhové prostředí (Common Language Runtime)
- bázová knihovna tříd (.NET Framework Class Library)
- společný typový systém (Commnon Type Syste)
- společná jazyková specifikace (Common Language Specification)
- programovací jazyky (Visual Basic, C#, Visual C++, J# a další)

Prvky vývojové platformy .NET Framework pracují společně pod operačními systémy řady Microsoft Windows. V této práci je rozebrána pouze malá část tohoto rozsáhlého rámce pro tvorbu aplikací. Jsou zde popsány pouze ty části, které umožní automatický převod aplikací z prostředí MS Access do .NET.

V prostředí .NET existuje celá řada typů aplikací jako jsou například: WPF, Windows Forms, ASP.NET nebo Console applications. MS Access aplikace se svým chováním a využitím nejvíce blíží aplikacím Windows Forms, proto byl tento typ aplikací zvolen pro zmíněný převod.

3.1 Ovládací prvky Windows Forms

Nejdůležitější třídou ve Windows Forms je `Control`. Z této třídy dědí všechny třídy ovládacích prvků jako např. tlačítko (`Button`), textové pole (`TextBox`), popisek (`Label`) a také samotný formulář (`Form`). [5]

Název	**Datový typ**	**Výchozí hodnota**	**Popis**
`Name`	`string`	`""`	Název, který se používá pro identifikaci komponenty v kódu.
`Left`	`int`	`0`	Umístění levého okraje komponenty.
`Top`	`int`	`0`	Umístění horního okraje komponenty.
`Width`	`int`	`0`	šířka
`Height`	`int`	`0`	výška
`BackColor`	`Color`	šedá	barva pozadí
`ForeColor`	`Color`	černá	barva písma
`Enabled`	`bool`	`true`	Nastavuje, zda je komponenta aktivní.
`Visible`	`bool`	`true`	Nastavuje, zda je komponenta viditelná.
`Text`	`string`	`true`	Text zobrazený na komponentě.

Pokračování tabulky na další stránce

Název	Datový typ	Výchozí hodnota	Popis
`Parent`	`Control`	`null`	Komponenta musí mít platného rodiče, má-li být na obrazovce vidět. Výjimku tvoří samotný formulář (`Form`), který má vlastnost `Parent` nastavenou na `null` a to znamená, že rodičem je pracovní plocha.
`Click`	`event`		Událost, která se vyvolá kliknutím na komponentu.

Tabulka 7: Seznam důležitých vlastností komponenty `Control`

3.1.1 Label

Popisek je ovládací prvek zobrazující uživateli neupravitelný text. Ten se zadává jako vlastnost `Text` příslušného ovládacího prvku.

3.1.2 LinkLabel

Tento ovládací prvek je odvozen od třídy `Label` a zobrazuje textový řetězec, který je jako celek nebo jako část odlišen a indikuje, že klepnutí na něj spustí nějakou akci, kupříkladu otevře určitou aplikaci nebo zobrazí webovou stránku.

Název	Datový typ	Výchozí hodnota	Popis
`LinkColor`	`Color`	modrá	Barva ve výchozím zobrazení.
`DisabledLinkColor`	`Color`	šedá	Barva neaktivního odkazu.
`ActiveLinkColor`	`Color`	červená	Barva aktivního odkazu.
`VisitedLinkColor`	`Color`	tmavě purpurová	Barva po použití odkazu.

Tabulka 8: Seznam důležitých vlastností komponenty `LinkLabel`

3.1.3 TextBox

Ovládací prvek `TextBox` je nejjednodušší formou prvku editace textu, je však dostatečně propracovaný na to, aby mohl být základem poznámkového bloku.

Název	Datový typ	Výchozí hodnota	Popis
`Multiline`	`bool`	`false`	Indikuje, zda může `TextBox` přijímat a zobrazovat více řádků textu.
`ReadOnly`	`bool`	`false`	Zobrazení neupravitelného textu.
`TextChanged`	`event`		Tato událost nastane, kdykoli se změní hodnota vlastnosti `Text`.

Tabulka 9: Seznam důležitých vlastností komponenty `TextBox`

3.1.4 CheckBox

`CheckBox` typicky představuje možnost výběru ze dvou možností `true` / `false`.

Název	Datový typ	Výchozí hodnota	Popis
`Checked`	`bool`	`false`	Indikuje, zda tlačítko aktuálně zobrazuje zaškrtnutí.
`ThreeState`	`bool`	`false`	třístavový `CheckBox`
`Appearance`	`Appearance`	`Normal`	Je-li nastavena vlastnost na `Appearance.Button`, pak se tento ovládací prvek bude podobat tlačítku, jenom bude přepínat svůj stav.
`AutoCheck`	`bool`	`true`	Je-li vlastnost nastavena na true, pak bude opakované klikání na políčko měnit stav `CheckState` v cyklu. V opačném případě se `CheckState` bude nastavovat manuálně.
`CheckedChanged`	`event`		Událost signalizuje změnu stavu zaškrtnutí.

Tabulka 10: Seznam důležitých vlastností komponenty `CheckBox`

3.1.5 RadioButton

`RadioButton` se používá pro výběr jedné z několika vzájemně se vylučujících možností.

Název	Datový typ	Výchozí hodnota	Popis
`Checked`	`bool`	`false`	Indikuje, zda tlačítko aktuálně zobrazuje zaškrtnutí.
`Appearance`	`Appearance`	`Normal`	Je-li nastavena vlastnost na `Appearance.Button`, pak se tento ovládací prvek bude podobat tlačítku, jenom bude přepínat svůj stav.
`AutoCheck`	`bool`	`true`	Pokud je vlastnost nastavena na `true` a `RadioButton` není právě zvolený, pak se klepnutím „zaškrtne“. Navíc dojde ke zrušení volby u všech ostatních prvků typu `RadioButton`, které mají společného rodiče.
`CheckedChanged`	`event`		Událost signalizuje změnu stavu zaškrtnutí.

Tabulka 11: Seznam důležitých vlastností komponenty `RadioButton`

3.1.6 ComboBox

Jedná se o rozbalovací seznam, který je podobný ovládacímu prvku `TextBox`, nicméně nabízí výběr z několika předdefinovaných hodnot. Seznam hodnot se zobrazí klepnutím na symbol (šipku) v pravé části prvku.

Název	Datový typ	Výchozí hodnota	Popis
`DropDowndStyle`	`ComboBoxStyle`	`DropDown`	`Simple` – Editovatelný text, seznam je vždy k dispozici. `DropDown` – Editovatelný text, seznam se rozevírá. `DropDownList` – Neupravitelný text, seznam se rozevírá.
`Items`	`ObjectCollection`	prázdná kolekce	Položky zobrazované v komponentě `ComboBox`. `ObjectCollection` definuje metodu `Add` pomocí níž lze přidat libovolný objekt do kolekce. Položky se zobrazují jako textové reprezentace, jak je nabízí metoda `ToString`.

Pokračování tabulky na další stránce

Název	Datový. typ	Výchozí hodnota	Popis
`SelectedItem`	`object`	`null`	Aktuálně vybraná položka.
`DataSource`	`object`	`null`	Vyplní seznam připojením datového zdroje. V takovém případě je nutné zadat vlastnosti `ValueMember` a `DisplayMember`.
`ValueMember`	`string`	`""`	Slouží pro specifikování názvu vlastnosti, která bude použita při získávání objektu pomocí vlastnosti `SelectedValue`.
`DisplayMember`	`string`	`""`	Slouží pro specifikování názvu vlastnosti, která bude zobrazena uživateli.
`SelectedIndex`	`int`	`-1`	Index aktuálně vybrané položky.
`SelectedIndex Changed`	`event`		Událost signalizuje, zda došlo ke změně výběru položky ze seznamu.
`TextChanged`	`event`		Událost signalizuje, zda došlo ke změně textu.

Tabulka 12: Seznam důležitých vlastností komponenty `ComboBox`

3.1.7 Button

Ovládací prvek je tlačítkem nejen proto, že na něj lze klepnout, ale především proto, že uživateli při klepnutí neboli stisku nabízí vizuální znázornění této akce.

3.1.8 GroupBox

Skupinový rámeček zobrazuje po svém obvodu linii, přičemž nahoře může být volitelně popisný text.

3.1.9 TabControl

`TabControl` umožňuje na formuláři vytvořit více navzájem překrývajících se karet. `TabControl` obsahuje kolekci panelů `TabPageCollection`, které představují jednotlivé karty v komponentě.

3.1.10 ToolStrip

`ToolStrip` je třída ve vývojovém rámci .NET Framework, která představuje panel nástrojů.

Tento ovládací prvek nabízí většinu flexibility a moderních funkcí, jaké jsou například v sadě Microsoft Office. Na `ToolStrip` panel lze umístit tyto komponenty: `Button`, `Label`, `SplitButton`, `DropDownButton`, `Separator`, `ComboBox`, `TextBox` a `ProgressBar`. Všechny tyto komponenty dědí ze třídy `ToolStripItem` (Tato třída není odvozena od `Control`, ale implementuje některé vlastnosti a události související s ovládacími prvky).

3.1.11 MenuStrip

Třída `MenuStrip` je odvozená od třídy `ToolStrip` a zobrazuje položky nejvyšší úrovně nabídky vodorovně přes celou horní část formuláře. Každá z takových položek nabídky, stejně jako všechny ostatní položky v nabídce, je samostatným objektem. Na `MenuStrip` lze umístit tyto komponenty: `MenuItem`, `ComboBox` a `TextBox`.

3.1.12 DataGridView

`DataGridView` často slouží jako kompletní řešení zobrazování nebo zadávání dat. Tento ovládací prvek uspořádává buňky do mřížky neboli tabulky řádků a sloupců. Ovládací prvek `DataGridView` je rozsáhle modifikovatelný, avšak v této práci je uvedeno pouze jednoduché využití.

`DataGridView` lze snadno napojit na datový zdroj pomocí vlastnosti `DataSource` a jako zdroj lze využít `DataSet`. Před nastavením vlastnosti `DataSource` ovládacího prvku `DataGridView` na objekt typu `BindingSource` je zapotřebí nastavit vlastnost `AutoGenerateColumns` na hodnotu `false`. Tím se zamezí vytváření sloupců pro jednotlivé vlastnosti vázaného zdroje. Pokud tato vlastnost bude nastavena na `true`, atributy tabulky se budou generovat automaticky a programátor nebude mít kontrolu nad výběrem a pořadí sloupců tabulky. Takové sloupce lze jednoduše vytvořit manuálně. Atributy tabulky jsou reprezentovány pomocí třídy `DataGridViewColumn`, ze které jsou dále odvozeny třídy pro reprezentování různých prvků v tabulce: [7]

- `DataGridViewButtonColumn` - tlačítko
- `DataGridViewCheckBoxColumn` - zaškrtávací tlačítko
- `DataGridViewComboBoxColumn` - rozevírací seznam
- `DataGridViewImageColumn` - obrázek
- `DataGridViewLinkColumn` - hypertextový odkaz
- `DataGridViewTextBoxColumn` - textové pole

4 Existující konverzní nástroje

4.1 Access Whiz

Access Whiz od společnosti Microtools [1] je nástroj pro převod MS Access aplikací. Ve svém balíčku nabízí jak převod do desktopových aplikací, tak i do webových. Dále podporuje několik programovacích jazyků jako jsou VB6, VB.NET a Visual C#. Pro připojení k databázi Access Whiz využívá technologii ADO.NET. Při použití tohoto nástroje lze převést formuláře a sestavy na webové formuláře a stále využívat i původní MS Access aplikaci. Cena kompletního zvýhodněného balíčku Access Whiz činí 390 $, ale společnost nabízí k prodeji i jednotlivé části, viz tabulka 13

Access Whiz	**Cena**
Access to VB .NET (Web Application)	$ 169.00
Access to VB .NET (Windows Application)	$ 169.00
Access to VB6 (Windows Application)	$ 94.00
Access to C# (Web Application)	$ 169.00
Access to Crystal Reports 8 or later	$ 169.00
Access Form Controls to VB .NET	$ 45.00
Access Form Controls to C# .NET	$ 45.00
Access to C# (Windows Application)	$ 169.00

Tabulka 13: Cena jednotlivých části nástroje Access Whiz k datu 20.3.2017

Na webových stránkách microtools.us je online k dispozici převedená aplikace Northwind [2] bez jakýchkoli úprav. Dále je zde umístěná i zkušební demo aplikace ke stáhnutí, kterou lze ovšem použít pouze pro převod aplikace Northwind. Při testování demo aplikace proběhl převod poměrně rychle (v rámci sekund) a převedl všechny formuláře. Formuláře, které byly vázány k datovému zdroji podporovaly i po převodu jednoduché CRUD operace nad databází. Tvůrce softwaru na svých webových stránkách uvádí, že složitější funkcionalita MS Access aplikace nebude převedena, což se potvrdilo i při testování.

[1] Microtools www.microtools.us
[2] Vzorová aplikace Northwind http://access-templates.com/tag/northwind.html

4.2 Evolution MS Access to VB.NET converter

Evolution MS Access to VB.NET converter vyvinula firma Redbrook Technology [3]. Tento nástroj dokáže převést jak nevázané, tak i vázané formuláře, karty na formuláři a další ovládací prvky. Pokud je na formuláři použit pro přístup k datům DAO (Data Access Object), je nástroj schopen převést tento kód do ADO.NET.

Na webových stránkách www.redbrooktech.com je k dipozici zkušební verze nástroje s možností převést až 10 formulářů. Při testování této zkušební verze program fungoval pouze s MS Access .mdb aplikací, kde se využívá Microsoft Jet Database Engine. Při volbě .adp aplikace připojené na SQL server, program vypíše chybové hlášení: "Odkaz není nastaven na instanci objektu". Při převádění aplikace Northwind se spustil MS Access a postupně se otevíraly převáděné formuláře. Několik formulářů se úspěšně podařilo převést, ovšem nikoli do spustitelného kódu. Cena tohoto nástroje činila k datu 20.3.2017 £ 145.

[3] Redbrook Technology http://www.redbrooktech.com

5 Komunikace se SŘBD

5.1 MS Access

Tato práce se zabývá konverzí MS Access aplikací využívajících nativního připojení k MS SQL Serveru. Nativní připojení k MS SQL Serveru z MS Access aplikace je realizováno pomocí komponenty ActiveX Data Objects (ADO), což je sada součástí umožňujících přístup k datům přes poskytovatele OLE DB (Object Linking and Embedding, Database).

V kapitole 5.1.1 je rozebráno, jakým způsobem komunikuje aplikace vytvořená pomocí MS Access s MS SQL Serverem. Ke každému testovacímu scénáři je uvedena akce MS Access aplikace, kterou se dotáže na SQL server. Tyto testy byly provedeny pomocí nástroje SQL Server Profiler, jenž je součástí instalace Microsoft SQL Serveru.

5.1.1 Komunikace na formuláři

Při otevření samostatného vázaného formuláře se nastaví (pomocí příkazu `SET ROWCOUNT[počet]`) počet záznamů, které mají být načteny z databáze. Tento údaj je nastavitelný ve vlastnostech formuláře (vlastnost *Maximální Počet Záznamů*) a nebo uživatelem v navigačním menu. Následuje samotný dotaz např.

```
SELECT * FROM "dbo"."Customers"
```

Pokud formulář obsahuje ovládací prvky, které vyžadují naplnění daty z databáze (např seznamy), aplikace se dotáže i na tyto dotazy, jež seznamy naplní.

Po naplnění formuláře daty z databáze lze mezi jednotlivými záznamy listovat bez toho, aniž by se aplikace musela pokaždé dotazovat na záznam v tabulce. Data jsou tedy uložena v paměti počítače, na kterém je MS Access aplikace spuštěna.

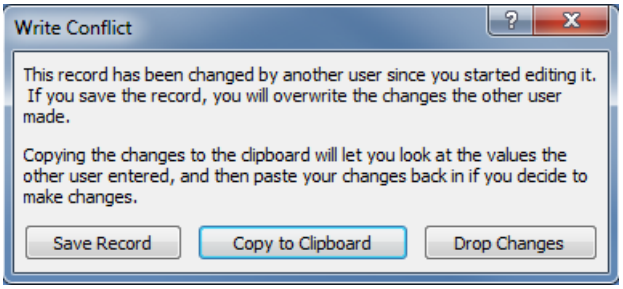
Pokud jsou na formuláři povoleny úpravy a hodnota některého ovládacího prvku byla od načtení změněna, záznam se zaktualizuje v době, kdy uživatel opustí aktuální záznam (např. při přechodu na jiný záznam, zavření formuláře a jiné). Aktualizace záznamu se provede pomocí dynamického SQL (procedura `sp_executesql`). Po provedení aktualizace se aplikace ještě dotáže na upravený záznam.

```
exec sp_executesql N'UPDATE "NorthwindCS"."dbo"."Customers" SET "City"=@P1
    WHERE "CustomerID"=@P2 AND "CompanyName"=@P3 AND "ContactName"=@P4 AND "
    ContactTitle"=@P5 AND "Address"=@P6 AND "City"=@P7 AND "Region" IS NULL AND
     "PostalCode"=@P8 AND "Country"=@P9 AND "Phone"=@P10 AND "Fax"=@P11',N'@P1
    varchar(9),@P2 varchar(5),@P3 varchar(23),@P4 varchar(10),@P5 varchar(20),
    @P6 varchar(14),@P7 varchar(8),@P8 varchar(5),@P9 varchar(7),@P10 varchar
    (10),@P11 varchar(10)','Stockholm','BLAUS','Blauer See Delikatessen','Hanna
     Moos','Sales Representative','Forsterstr. 57','Mannheim','68306','Germany'
    ,'0621-08460','0621-08924'
```

```
exec sp_executesql N'SELECT * FROM ( SELECT * FROM "dbo"."Customers" ) AS
    DRVD_TBL WHERE "CustomerID" = @P1',N'@P1 varchar(5)','BLAUS'
```

Výpis 1: Editace záznamu

Pokud byl záznam mezi načtením a editováním změněn třetí stranou, například jiným uživatelem pracujícím nad stejnými daty, MS Access tuto kolizi rozpozná a nabídne uživateli možnost přepsání záznamu nebo ponechání změn.



Obrázek 5: Kolize při úpravě záznamu

Při vkládání nového záznamu se před vyplněním formuláře nejprve získá struktura tabulky:

```
EXEC sp_MShelpcolumns N'Customers', NULL, N'id', 1
```

Výpis 2: Získání informace o tabulce Customers

Uživatel formulář vyplní a vložení do databáze se provede po opuštění záznamu (např. při přechodu na jiný záznam, zavření formuláře a jiné).

```
exec sp_executesql N'INSERT INTO "NorthwindCS"."dbo"."Customers" ("CustomerID
    ","CompanyName","ContactName","ContactTitle","Address","City","PostalCode
    ","Phone") VALUES (@P1,@P2,@P3,@P4,@P5,@P6,@P7,@P8)',N'@P1 varchar(5),@P2
    varchar(21),@P3 varchar(15),@P4 varchar(20),@P5 varchar(29),@P6 varchar(6),
    @P7 varchar(7),@P8 varchar(14)','CNSHO','Consolidated Holdings','Elizabeth
    Brown','Sales Representative','Berkeley Gardens
12  Brewery','London','WX1 6LT','(171) 555-2282'

exec sp_executesql N'SELECT * FROM ( SELECT * FROM "dbo"."Customers" ) AS
    DRVD_TBL WHERE "CustomerID" = @P1',N'@P1 varchar(5)','CNSHO'
```

Výpis 3: Vložení záznamu

```
exec sp_executesql N'DELETE FROM "NorthwindCS"."dbo"."Categories" WHERE "
    CategoryID"=@P1 AND "CategoryName"=@P2',N'@P1 int,@P2 varchar(10)',22,'
    Cereals UK'
```

Výpis 4: Odstranění záznamu

Pokud se bude jednat o datový list (viz kapitola 2.1.2), komunikace s databází bude obdobná.

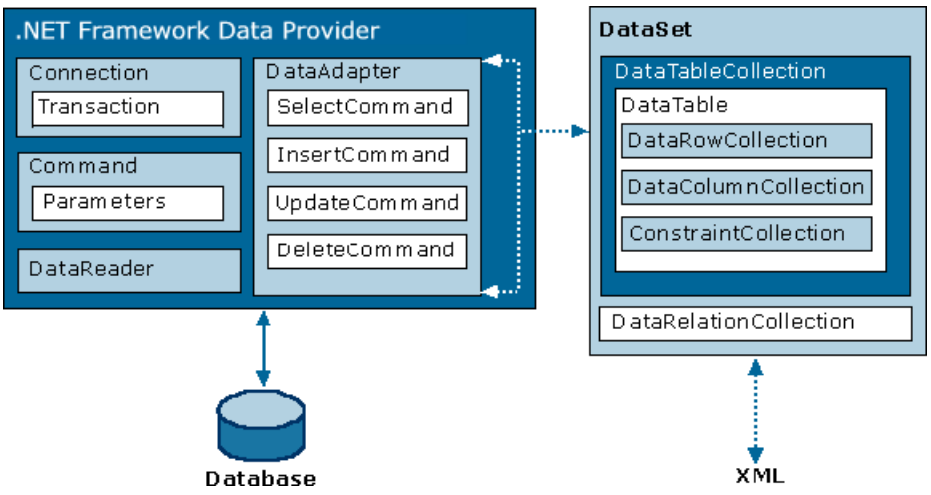
5.2 .NET Framework

5.2.1 ADO.NET

ADO.NET je rozsáhlou kolekcí tříd, které umožňují pracovat s databázemi a datovými soubory v aplikačním rámci .NET.

Kolekce tříd ADO.NET byly navrženy tak, aby se oddělil způsob přístupu k datům od manipulace s daty. ADO.NET používá vícevrstvou architekturu založenou na poskytovatelích dat (Data Provider), což je sada tříd, které jsou určeny pro přístup ke konkrétní databázi, vykonávání SQL příkazů a získávání dat. [8] Do těchto tříd patří:

- `Connection` – Poskytuje spojení se zdrojem dat.
- `Command` – Vykonává SQL příkazy.
- `DataReader` – Poskytuje rychlý přístup k datům získaným z dotazu.
- `DataAdapter` – Naplní sadu dat (`DataSet`) a promítne změny do zdroje dat



Obrázek 6: Architektura ADO.NET

5.2.2 Objektově relační mapování

Při modelování a vývoji aplikací je snaha co nejvěrněji zachytit realitu. Objekty reálného světa jsou v aplikaci reprezentovány jako entity. Zatímco je v relační databázi entita reprezentována jako záznam, resp. množina záznamů v databázových tabulkách, tak v objektově orientovaném jazyce je entita zpravidla reprezentována jako instance nějaké třídy. Tato rozdílná reprezentace entit vedla ke vzniku programovací techniky označované jako objektově relační mapování (ORM). Tato technika slouží pro propojení objektově orientovaného programovacího jazyka s relační strukturou databáze, tzn. snaží se dát vývojáři unifikovaný přístup k libovolnému objektu, resp. entitě, se kterou v aplikaci pracuje. Díky tomu je vývojář při práci s daty, které jsou v aplikaci reprezentovány právě pomocí objektů, do jisté míry odstíněn od nutnosti pracovat s SQL dotazy konkrétní relační databáze. Použití ORM usnadňuje zejména provádění jednoduchých CRUD operací jako jsou čtení, zápis, úprava a mazání dat. ORM se dále stará o automatickou konverzi rozdílných datových typů mezi databázovým systémem a programovacím jazykem. [9]

Pro .NET Framework existuje celá řada objektově relačních mapování, z nichž jsou dvě implementace přímo součástí samotného rámce .NET a to ADO.NET Entity Framework a LINQ to SQL. Objektově relační mapování si můžeme naprogramovat i svoje vlastní přesně pro potřeby dané aplikace, čímž získáme stoprocentní kontrolu nad všemi SQL operacemi.

6 Programový přístup k aplikaci MS Access

Při konverzi aplikace MS Access je nejprve nutné zjistit, z jakých objektů se aplikace skládá a jak jsou objekty navzájem propojeny. Pro automatickou konverzi je tedy nezbytné programově analyzovat strukturu samotné aplikace. To je možné provést třemi způsoby: (1) analýzou souboru aplikace ADP, (2) použitím knihovny pro manipulaci s aplikací MS Access z prostředí .NET a (3) analýzou samotné aplikace kódem VBA pomocí reflexe. Vzhledem k tomu, že ADP je proprietární souborový formát, první možnost nepřipadá v úvahu. Druhá zmiňovaná možnost představuje použití knihovny Microsoft Access Object Library[4], která mapuje objekty v aplikaci MS Access na objekty .NET. Nevýhodou této knihovny je ovšem její pomalá odezva. Poslední možností je vytvoření modulu v jazyce VBA, který bude exportovat strukturu aplikace do externího souboru, přičemž strukturu bude zjišťovat za pomocí reflexe. Tato třetí možnost byla nakonec využita pro automatickou konverzi.

6.1 Modul pro export vlastností MS Access aplikace

Pro export všech objektů včetně jejich vlastností, které jsou důležité z hlediska automatické konverze, byl v rámci této práce vytvořen jednoduchý modul VBA, obsažený v přiloženém souboru `AccessModule.txt`. Modul se skládá z procedury `Export`, která přistupuje ke všem objektům, jako jsou makra, moduly, formuláře atd. pomocí vlastnosti `Application.CurrentProject`. Při spuštění této procedury se vytvoří složka `AccessExportedData` v uživateli definovaném adresáři a do ní se vloží soubory:

- `ObjectDB.txt` – Obsahuje seznam vlastností aplikace včetně vlastností připojení k databázi na SQL Serveru. Pokud je v MS Access aplikaci použita SQL autentizace a přihlašovací údaje jsou uloženy v aplikaci, budou tyto údaje (včetně hesla) uloženy jako běžný text. Ukázku části procedury `Export`, která tyto vlastnosti prochází v cyklu a vepisuje je do souboru `ObjectDB.txt`, vidíme ve výpisu 5. Funkce `Check` kontroluje, zda se v hodnotách vlastností nevyskytuje zakázaný znak jako například středník nebo odřádkování. Tyto znaky slouží jako oddělovače v exportovaném souboru, a proto jsou nahrazovány řetězcem `##Chr(59)##` pro středník a odřádkování řetězcem `##Chr(13)##`.

[4]MSDN https://msdn.microsoft.com/en-us/library/15s06t57.aspx

```
Set dbs = Application.CurrentProject
File = RootFile & "\ObjectDB.txt"
'vlastnosti projektu
Open (File) For Output As #1
For i = 0 To dbs.Properties.Count - 1
   Value = dbs.Properties(i).Value
   Value = Check(Value)
   Print #1, "CurrentProject;" & dbs.Properties(i).Name & ";" & Value
Next i

'vlastnosti pripojeni
For i = 0 To dbs.Connection.Properties.Count - 1
   Value = dbs.Connection.Properties(i).Value
    Value = Check(Value)
    Print #1, "Connection;" & dbs.Connection.Properties(i).Name & ";" &
       Value
Next i
```

Výpis 5: Export vlastností aplikace

Pro každou vlastnost z kolekce `Application.CurrentProject.Properties` bude v souboru vytvořen samostatný řádek se strukturou:

`CurrentProject;`*`Název vlastnosti`*`;`*`Hodnota`*

Pro každou vlastnost z kolekce `Application.CurrentProject.Connection.Properties` bude vytvořen řádek:

`Connection;`*`Název vlastnosti`*`;`*`Hodnota`*

- `ObjectForm.txt` – Procedura `Export` dále postupně otevírá všechny formuláře v aplikaci a zapisuje do souboru `ObjectForm.txt` všechny jeho vlastnosti, objekty na něm obsažené a vlastnosti těchto objektů. Pro každou vlastnost je vytvořen řádek s následující strukturou:

  `Form;`*`Název Formuláře`*`;`*`Typ objektu`*`;`*`Název objektu`*`;`*`Název vlastnosti`*`;`*`Hodnota`*

  V souboru bude tedy například uloženo:

  ```
  ...
  Form;FPenalty_sub;acTextBox;Notification;Visible;True
  Form;FPenalty_sub;acTextBox;Notification;Width;1701
  Form;FPenalty_sub;acTextBox;Notification;Height;240
  ...
  ```

Což znamená, že na formuláři s názvem *FPenalty_sub* je obsaženo textové pole *Notification*, které má vlastnost *Visible* nastavenou na *True*. Dále je zde definovaná šířka a výška tohoto textového pole.

Typ objektu je ve vlastnostech formuláře uložen jako číselná hodnota. Aby byl obsah exportovaného souboru samopopisný, obsahuje modul `AccessModule` funkci `ControlTypeTxt`, která přijímá číselný typ objektu a vrací odpovídající textovou popisnou hodnotu. Ve výpisu 6 je vidět ukázka této funkce pouze se třemi typy hodnot.

```
Function ControlTypeTxt(Typ As Integer) As String
    Select Case Typ
        Case 100
            ControlTypeTxt = "acLabel"
        Case 112
            ControlTypeTxt = "acSubform"
        Case 109
            ControlTypeTxt = "acTextBox"

        '...

        Case Else
            ControlTypeTxt = CStr(Typ)
    End Select
Exit Function
```

Výpis 6: Funkce `ControlTypeTxt`

- `ObjectMenu.txt` – Jednotlivé záložky hlavní nabídky jsou obsaženy v souboru `ObjectMenu.txt`, kde jsou zapsány jejich vlastnosti i s odkazy na případné položky. Tyto položky jsou spouštěny jako jednotlivá makra, takže budou uložena v adresáři `Macros`. Pro každou položku z nabídky je v souboru vygenerován řádek se strukturou:

  `Menu;`*`Název menu`*`;`*`Název podmenu`*`;`*`Hodnota`*

  V souboru bude tedy například uloženo:

  ```
  ...
  Menu;Lists;Penalty;OnAction;Menu.Penalty
  Menu;Lists;Penalty;Enabled;True
  ...
  ```

  První řádek v tomto příkladu znamená, že v hlavní nabídce záložka *Lists* obsahuje položku *Penalty*, která spouští makro s názvem *Penalty*. Toto makro můžeme vidět ve výpisu 7.

Druhý řádek definuje položce *Penalty* vlastnost *Enabled*, která značí zda bude tato položka aktivní. V tomto případě je tato hodnota nastavena na *True*.

- `Modules_G` – Adresář obsahující všechny globální moduly, které byly v aplikaci obsaženy. Pro každý modul VBA se vytvoří samostatný soubor s názvem modulu, kde bude uložen jeho zdrojový kód.
- `Modules_F` – Adresář obsahující moduly formulářů. Stejně jako u globálních modulů bude mít každý formulář vytvořený jeden soubor.
- `Macros` – Adresář obsahující všechna makra aplikace. Obsahuje také speciální soubor `Menu.txt`, kde jsou uložena makra pro ovládání hlavního menu. Příklad takového makra je vidět ve výpisu 7.

```
Begin
    MacroName ="Penalty"
    Action ="OpenForm"
    Argument ="FPenalty"
    Argument ="0"
    Argument =""
    Argument =""
    Argument ="-1"
    Argument ="0"
End
```

Výpis 7: Příklad makra spouštějící formulář *FPenalty*

7 Použité knihovny

Tato kapitola je zaměřena na stručný popis knihoven, které jsou využívány v prototypu implementované konverzní aplikace. Tyto knihovny poskytují funkcionalitu, která není standardně obsažena v rámci .NET.

7.1 SmartISLib

Knihovna SmartISLib vyvinutá společností NATIVA Enterprise s.r.o.[5] automatizuje některé činnosti aplikace informačního systému související zejména se standardními CRUD operacemi. Pro konverzi MS Access aplikací se z této knihovny využívá objektově-relační mapování, dělení aplikace do modulů a provázání prvků uživatelského rozhraní s datovou vrstvou. [6]

7.1.1 Rozdělení aplikace do modulů

Knihovna je založena na myšlence, že aplikaci je možné rozdělit do množství modulů, které jsou na sobě navzájem nezávislé. Každý modul lze spustit v několika instancích. Spuštění modulu vyžaduje *spouštěcí parametry*, které mohou novou instanci nějakým způsobem ovlivnit (např. otevřít detail záznamu s určitou hodnotou primárního klíče). Knihovna rozlišuje několik typů modulů, kde každý z nich definuje určité specifické chování. Používání modulů zajišťuje konzistentní chování uživatelského rozhraní, které výrazně usnadňuje orientaci v aplikaci informačního systému. Typy modulů jsou následující:

1. *Samostatný modul* (třída `StandAloneModule`) – tento typ vyžaduje implementaci jediné metody, která je spuštěna při startu modulu. Jde o nejjednodušší typ modulu, který poskytuje programátorovi největší volnost při implementaci. Na druhou stranu ale nedokáže zajistit konzistentní chování napříč vyvíjenou aplikací. Tento typ se hodí pro řešení samostatných speciálních formulářů např. pro nastavení chování aplikace informačního systému.

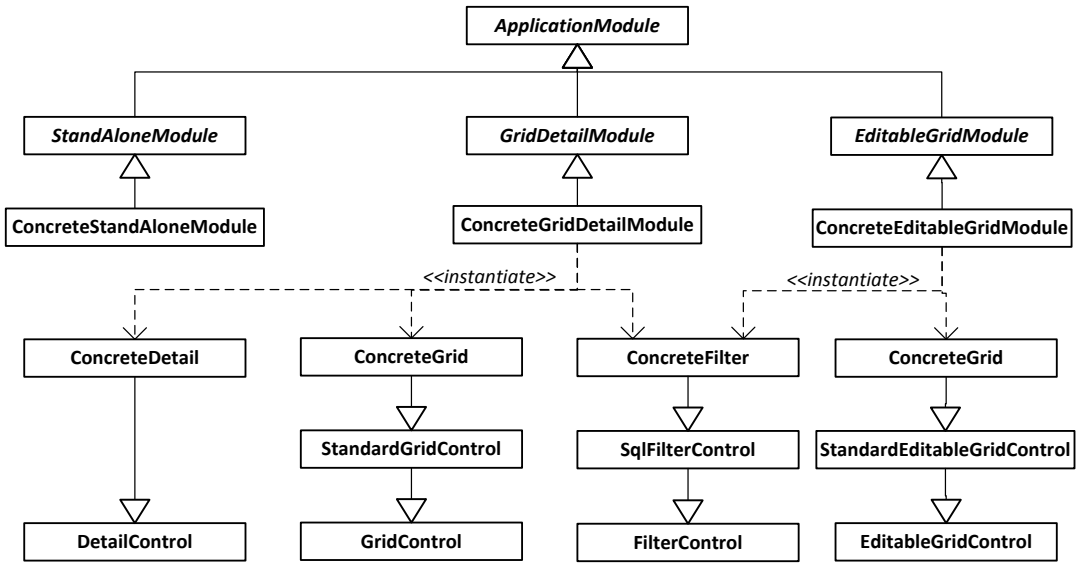
2. *Tabulkové zobrazení + detail* (třída `GridDetailModule`) – jde o nejběžnější typ modulu, který lze použít pro CRUD operace nad tabulkou nebo skupinou tabulek, které tvoří jeden logický celek. Tento typ definuje několik uživatelských komponent, které je potřeba implementovat: *tabulkové zobrazení*, *filtr* a *detail*. Komponenty není nutné implementovat všechny, modul může např. obsahovat jen tabulkové zobrazení s filtrem. Všechny tři komponenty budou rozebrány dále v této kapitole. Moduly tohoto typu se zobrazují na kartách hlavního formuláře aplikace.

[5] NATIVA Enterprise s.r.o. http://www.nativa.cz

[6] Autor práce se společností NATIVA Enterprise s.r.o. spolupracuje a na vývoji knihovny se z velké části sám podílel. Společnost souhlasila s využitím knihovny v této diplomové práci.

3. *Editovatelné tabulkové zobrazení* (třída `EditableGridModule`) – tento typ modulu se používá pro jednoduchou editaci tabulek. Stejně jako u předchozího je potřeba implementovat *tabulkové zobrazení* a volitelně pak *filtr*. *Detail* v tomto typu modulu implementovat nelze a data se editují přímo v tabulkovém zobrazení. Moduly tohoto typu se také zobrazují na kartách hlavního formuláře aplikace.

Třídní diagram zachycující zjednodušenou statickou strukturu modulů můžeme vidět na obrázku 7. Třídy s prefixem `Concrete` představují příklady implementace ostatních abstraktních tříd. Význam jednotlivých tříd je rozebrán v následujících podkapitolách.



Obrázek 7: Moduly knihovny SmartISLib

7.1.2 Tabulkové zobrazení (třída `GridControl`)

Tabulkové zobrazení je komponenta vyžadující implementaci metody `RefreshData`, která se automaticky volá při spuštění modulu a při kliknutí na tlačítko pro obnovení dat. Samotná třída `GridControl` je odvozena od třídy `UserCotrol`, která tvoří standardní součást rámce .NET. Třídu je tedy možné doplnit o libovolné ovládací prvky uživatelského rozhraní a to pomocí návrhového zobrazení ve vývojovém nástroji Microsoft Visual Studio. Pro realizaci tabulkového zobrazení umožňuje knihovna implementovat přímo třídu `GridControl`, která neklade žádné omezení, jak má tabulkové zobrazení vypadat, nicméně jednodušší možností obvykle je implementovat třídu `StandardGridControl` odvozenou od `GridControl`, která předepisuje použití standardní .NET komponenty `DataGridView` (viz kapitola 3.1.12). Třída `StandardGridControl` vyžaduje nastavení vlastností `SqlSelect` a `PrimaryKeyAttribute`, kde první z uvedených vlastností představuje SQL dotaz na databázi a druhá udává, který z navrácených atributů představuje primární klíč. Nastavení vlastnosti `PrimaryKeyAttribute` umožní automatické otevření

detailu po dvojkliku na řádek v komponentě `DataGridView`. Nastavením vlastností `SqlSelect` a `PrimaryKeyAttribute` a nadefinováním sloupců v komponentě `DataGridView` je tabulkové zobrazení pro obvyklé případy hotovo a není nutné doplňovat žádný programový kód. Je-li součástí modulu také implementace filtru, bude filtr pomocí klauzule `WHERE` omezovat výsledek dotazu nastaveném ve vlastnosti `SqlSelect`.

7.1.3 Editovatelné tabulkové zobrazení (třída `EditableGridControl`)

Třída `EditableGridControl` je určená pro použití u editovatelného tabulkového zobrazení. Obsahuje virtuální metody jako `NewRecordCore`, `DeleteRecordCore` a `SaveDataCore`, které je nutné implementovat pro editování záznamů v tabulce. Takové řešení je obsaženo ve třídě `StandardEditableGridControl`, která je odvozená z `EditableGridControl`. Při použití tohoto kompletního řešení editování záznamů v tabulce je vyžadováno nastavení vlastností `SqlSelect` a `PrimaryKeyAttribute`.

7.1.4 Filtr (třída `FilterControl`)

Účelem filtru je sestavit podmínku `WHERE` pro SQL dotaz uvedený ve vlastnosti `SqlSelect` v tabulkovém zobrazení `StandardGridControl`. Filtr předepisuje metodu `GetSqlWhere`, která vrací podmínku v podobě textového řetězce. Knihovna nabízí standardní implementaci filtru v podobě třídy `SqlFilterControl`, kterou je možné velmi jednoduše nastavit přímo v návrhovém zobrazení. Standardní filtr se skládá z množiny filtrovacích položek, které mohou být různého typu, např.: `StringFilterItem` pro filtrování polí s textovými řetězci, `IntegerFilterItem` pro filtrování polí s čísly, `DateFilterItem` pro filtrování datumových polí, `BoolFilterItem` pro filtrování polí typu `BIT` nebo `IntForeignKeyFilterControl` pro filtrování pomocí rozbalovací nabídky, která obsahuje hodnoty z číselníku. Jednotlivým položkám je obvykle nutné pouze nastavit vlastnost `Attribute`, která ve formě textového řetězce představuje filtrovaný atribut, a `Title`, která reprezentuje lidsky čitelný popis atributu zobrazený vedle filtrovacího pole.

7.1.5 Detail (třída `DetailControl`)

Tato třída je opět odvozena od `UserControl`. Detail záznamu je zobrazen, je-li modul spuštěn se speciálními spouštěcími parametry. Pomocí spouštěcích parametrů lze také ovlivnit, zda bude detail připraven pro založení nového záznamu nebo načte již existující záznamy. Detail neslouží vždy k editaci jediného záznamu, je možné načíst také např. související záznamy ve vztahu 1:N, které budou zobrazeny a upravovány pomocí komponenty `DataGridView`. Pro funkčnost detailu je nutné naimplementovat následující metody:

1. `PrepareNew` – metoda je spuštěna, je-li detail otevřen pro vytvoření nového záznamu. V této metodě je obvykle nachystána jedna nebo více nových instancí objektů objektově-relačního mapování.

2. `LoadRecord` – metoda je spuštěna, je-li detail otevřen pro zobrazení a editaci existujícího záznamu. V této metodě je obvykle načten jeden nebo více objektů objektově-relačního mapování.

3. `BindData` – metoda je spuštěna po `PrepareNew` nebo `LoadRecord` a slouží především k provázání objektů objektově-relačního mapování s prvky uživatelského rozhraní, které se do detailu umístí v návrhovém zobrazení komponenty. Třída `DetailControl` poskytuje množství metod, kterými je možné jednoduše svázat atribut objektu objektově-relačního mapování se standardním ovládacím prvkem jako je např. textové pole nebo zaškrtávací tlačítko. Metodám je předán objekt reprezentující ovládací prvek, objekt objektově-relačního mapování a název atributu tohoto objektu. Ukázku volání těchto metod vidíme ve výpisu 8, kde je provázáno textové pole `txbInvoiceNumber` s atributem `Number` objektu `invoice` a zaškrtávací tlačítko `cbRoundTotal`, které je provázáno se stejným objektem a atributem `RoundTotal`.

```
protected override void BindData()
{
   BindTextBox(txbInvoiceNumber, invoice, "Number");
   BindCheckBox(cbRoundTotal, invoice, "RoundTotal");
}
```

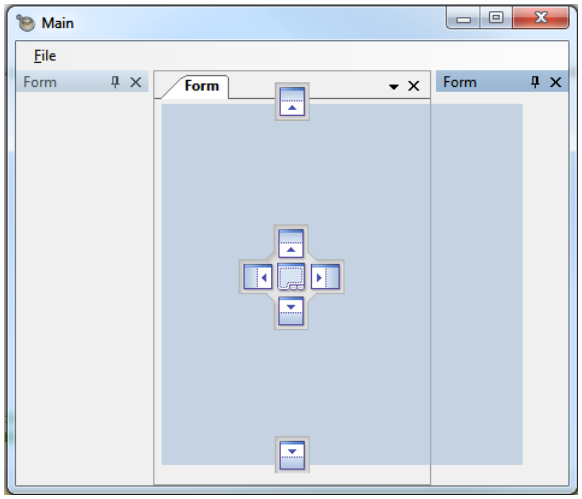
Výpis 8: Ukázka obsahu metody `BindData`

4. `InsertRecord`, `UpdateRecord`, `DeleteRecord` – jak již z názvu jednotlivých metod vyplývá, metody slouží k vložení, aktualizaci nebo smazání záznamů z databáze prostřednictvím objektově-relačního mapování.

7.2 WeiFen Luo's DockPanelSuite

Uživatelské rozhraní moderních desktopových aplikací je často založeno na používání záložek (tzv. *karet*). Příkladem takové aplikace je třeba samotný nástroj Microsoft Visual Studio. Uživatelské rozhraní aplikací vytvořených prototypem navrhovaného konverzního nástroje využívají karty pro tabulková zobrazení jednotlivých modulů. Funkcionalitu karet zajišťuje knihovna WeiFen Luo's DockPanelSuite[7]. Existuje celá řada podobných řešení, ale většina z nich není volně dostupná, takže WeiFen Luo's DockPanelSuite je ideální volbou. Jediným nedostatkem je chybějící podrobná dokumentace, nicméně použití této knihovny je velice jednoduché a knihovna obsahuje jen minimální množství chyb.

[7] DockPanel Suite http://dockpanelsuite.com



Obrázek 8: Ukázka uživatelského rozhraní s použitím knihovny WeiFen Luo's DockPanelSuite

7.3 SmartORMGen

Tato knihovna je na rozdíl od ostatních využívána při samotné konverzi. Jedná se o generátor objektově relačního mapování vyvinuté pro aplikace využívající knihovnu SmartISLib. Tato knihovna byla také vyvinutá společností NATIVA Enterprise.

8 Implementace aplikace pro automatickou konverzi

Aplikace MS Access se se svými formuláři nejvíce podobají aplikacím typu Windows Forms v aplikačním rámci .NET a právě této podobnosti bylo při implementaci využito. Konečná převedená aplikace tedy představuje desktopovou aplikaci typu Windows Forms a její zdrojové kódy jsou generovány v jazyce C#. Implementovaný nástroj Access Converter je zaměřen především na konverzi uživatelských formulářů. Pro konverzi tiskových sestav je možné použít postup, který je popsán v kapitole 9. Moduly psané v jazyce VBA a makra jsou při konverzi vypuštěny – tuto funkcionalitu je tedy nutné po provedení konverze doprogramovat. Samotná aplikace Access Converter je také vyvinuta v prostředí .NET v jazykce C#.

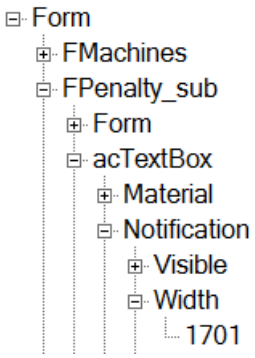
Aplikační rámec Microsoft .NET je navržen velmi obecně. Standardní činnosti aplikací informačního systému jako je např. provázání ovládacích prvků s atributy tabulky nebo uživatelsky pohodlné filtrování tabulkového zobrazení v tomto rámci samozřejmě je možné implementovat, avšak vyžaduje to mnohem hlubší znalosti vývojáře než v případě aplikace MS Access. Existují však knihovny, které tyto činnosti v prostředí .NET částečně automatizují. Pro tyto účely je využita knihovna SmartISLib popsaná v kapitole 7.1.

Princip činnosti navrhovaného nástroje je možné shrnout do 4 kroků: (1) parsování vstupních souborů, (2) detekce a příprava modulů, (3) generování kódu vybraných modulů a (4) generování ORM. Převod nebude proveden plně automaticky, ale za pomoci průvodce s grafickým uživatelským rozhraním, kterým bude možné jednotlivé kroky ovlivnit. Jednotlivé kroky budou rozebrány v kapitole 8.1. Kapitola 8.2 pak představí implementovanou aplikaci z hlediska uživatele, který bude převod provádět.

8.1 Princip převodu

8.1.1 Parsování vstupních souborů

Vstupem do aplikace Access Converter jsou kolekce vlastností, které generuje modul popsaný v kapitole 6.1. Pro načtení těchto souborů byla vytvořena třída `AccessDataLoader`, která soubory `ObjectDB.txt`, `ObjectMenu.txt` a `ObjectForm.txt` převede na stromovou strukturu, kde jsou jednotlivé hodnoty vlastností uloženy v listových uzlech. Pro reprezentaci stromu byla použita třída `TreeNode` ze jmenného prostoru `System.Windows.Forms`. Stromovou strukturu realizovanou pomocí objektů typu `TreeNode` je možné jednoduše nastavit jako vlastnost standardní komponenty `TreeView`. Ukázku stromu vidíme na obrázku 9.



Obrázek 9: Ukázka stromové struktury v komponentě `TreeView`

8.1.2 Detekce a příprava modulů

V této fázi konverzní nástroj seskupí formuláře aplikace MS Access (dle stromové struktury vytvořené ze souboru `ObjectForm.txt`) do modulů. Ty jsou detekovány na základě struktuy hlavní nabídky (dle stromové struktury vygenerované ze souboru `ObjectMenu.txt`). Pokud je u těchto položek přiřazeno makro (vlastnost *OnAction*), načte se soubor `Menu.txt` umístěný v adresáři `Macros`. V tomto souboru jsou uložena makra definující akci, která nastane při kliknutí na položku v menu. Pokud je touto akcí otevření formuláře, pak se podle tohoto formuláře automaticky vytvoří modul (viz kapitola 7.1) podle následujících pravidel:

- Pokud je v menu spouštěn editovatelný datový list (viz kapitola 2.1.2), vytvoří se modul obsahující tabulkové zobrazení odvozené od třídy `StandardEditableGridControl` (viz kapitola 7.1). V případě needitovatelného datového listu bude tabulkové zobrazení dědit ze třídy `StandardGridControl`.
- Pokud se spouští formulář, který má vnořený datový list, vytvoří se modul obsahující tabulkové zobrazení s přidruženým filtrem (třída dědící z `SqlFilterControl`, viz kapitola 7.1). Datový list se převede na tabulkové zobrazení jako v předchozím bodě a ovládací prvky obsažené na spouštěném formuláři se transformují na filtrovací elementy.
- Pokud není splněna ani jedna z předchozích podmínek, formulář se převede na obyčejný formulář dědící ze třídy `Form` bez jakékoli funkcionality. Převedeno je tedy pouze samotné uživatelské rozhraní.

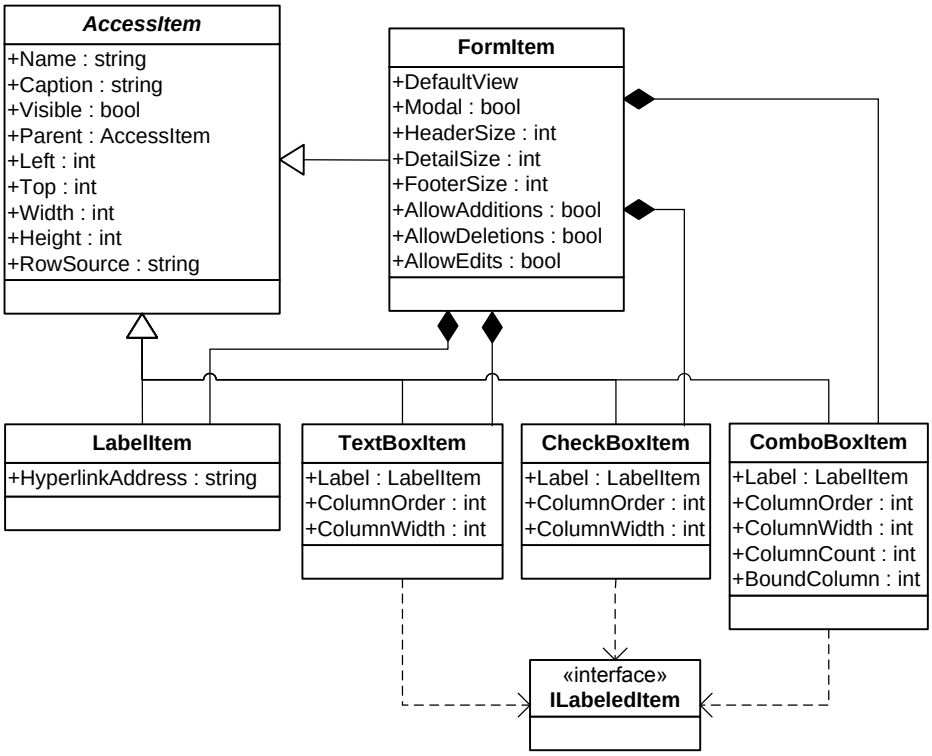
Jestliže převáděná aplikace žádné hlavní menu neobsahuje, nebo obsahuje menu vytvořené pomocí *Ribbonů* (viz kapitola 2.3), žádné moduly se automaticky nevytvoří a je jen na uživateli, jaké moduly si nadefinuje v průvodci nastavením aplikace Access Converter.

8.1.3 Generování kódu vybraných formulářů

Před generováním kódu vybraných formulářů je potřeba sestavit jejich model. Na základě modelu jsou pak generovány zdrojové soubory v jazyce C#.

8.1.3.1 Vytvoření modelu formuláře Model se vytváří na základě stromové struktury ze souboru `ObjectForm.txt`. Tento soubor obsahuje všechny formuláře původní aplikace a jejich vlastnosti. Kořenem vytvořené stromové struktury je *Form* a všichni přímí potomci představují jednotlivé formuláře. Tyto uzly dále obsahují podstrom s vlastnostmi samotného formuláře a podstromy všech obsažených prvků na něm. Tyto prvky mají samozřejmě také své vlastnosti a ty budou obsaženy v jejich podstromech.

Zjednodušená struktura tříd modelu je zachycena na obrázku 10. Třídy modelu obsahují vlastnosti odpovídajících ovládacích prvků aplikace MS Access, které byly popsány v kapitole 2.2. Všechny tyto třídy dědí ze třídy `AccessItem`, která obsahuje společné vlastnosti všech ovládacích prvků. Speciálním případem prvku je také `FormItem` (formulář), který obsahuje seznamy všech ovládacích prvků obsažených na formuláři.



Obrázek 10: Model formulářů aplikace MS Access

8.1.3.2 Generování souborů Formulář MS Access je do aplikace .NET převeden na třídu odvozenou od jednoho z následujících typů: `StandardGridControl`, `StandardEditableGridControl` (viz kapitola 8.1.3.3), `SqlFilterControl` (viz kapitola 8.1.3.4), `DetailControl` (viz kapitola 8.1.3.5), `FormMain` a `Form`. Všechny třídy s výjimkou `Form` jsou obsaženy v knihovně SmartISLib (viz kapitola 7.1). Volba typu závisí na typu modulu (viz kapitola 8.1.2). Jelikož všechny uvedené třídy dědí ze standardních .NET tříd `Form` nebo `UserControl`, je možné využívat návrhové zobrazení nástroje Microsoft Visual Studio. Pro každou vytvořenou

třídu je proto nutné vygenerovat dva soubory: `.cs` obsahující zdrojový kód a `.designer.cs` obsahující design. Oba soubory tvoří jednu parciální třídu. Implementovaný nástroj tedy musí kód uživatelského rozhraní generovat stejnou formou jako Visual Studio. Pro generování formulářů byly vytvořeny tyto třídy:

- `GenerateFormMain` – Generuje hlavní formulář (`FormMain`), který bude zobrazován při startu převedené aplikace. Na tomto formuláři bude v horní části umístěno hlavní menu (`MenuStrip` viz kapitola 3.1.11), které bude spouštět vygenerované moduly a formuláře. Pod touto nabídkou bude umístěn panel nástrojů (`ToolStrip` viz kapitola 3.1.10), který bude viditelný pouze v případě, že bude spuštěný nějaký modul.

  V dolní části hlavního formuláře bude umístěný `StatusStrip`, na kterém bude obsažen jeden `ToolStripStatusLabel`, zobrazující jméno uživatele připojeného k MS SQL Serveru.

- `GenerateFormCode` – Generuje soubor `.cs` pro klasické formuláře odvozené ze třídy `Form`.

- `GenerateDetailFiles` – Generuje soubor `.cs` pro detaily modulu odvozené ze třídy `DetailControl`.

- `GenerateFormDesigner` – Generuje soubor `.Designer.cs` pro klasické formuláře a pro detaily modulu.

- `GenerateGridFiles` – Generuje soubor `.cs` pro tabulkové zobrazení a XML soubor zdrojů `.resx`.

- `GenerateGridDesigner` – Generuje soubor `.Designer.cs` pro tabulkové zobrazení.

- `GenerateFilterFiles` – Generuje soubory filtru.

Visual Studio navíc generuje XML soubor zdrojů `.resx`, který obsahuje doplňující informace o formuláři. Například při vytváření objektu typu `DataGridView` (viz kapitola 8.1.3.3) tento soubor obsahuje doplňující informace o jednotlivých sloupcích. Aby bylo možné po konverzi používat návrhový režim v nástroji Visual Studio, musí konverzní aplikace takový soubor vygenerovat. Příklad vygenerovaného souboru je vidět ve výpisu 9.

```
<?xml version="1.0" encoding="utf-8"?>
<root>
  <resheader name="resmimetype">
    <value>text/microsoft-resx</value>
  </resheader>
  <resheader name="reader">
    <value>System.Resources.ResXResourceReader, System.Windows.Forms</value>
  </resheader>
  <resheader name="writer">
```

```
    <value>System.Resources.ResXResourceWriter, System.Windows.Forms</value>
  </resheader>
  <data name="txbCountryCode.UserAddedColumn" type="System.Boolean, mscorlib">
    <value>True</value>
  </data>
  <data name="txbCountry.UserAddedColumn" type="System.Boolean, mscorlib">
    <value>True</value>
  </data>
</root>
```

Výpis 9: Příklad vygenerovaného XML souboru zdrojů

Během generování zdrojových souborů je nutné přepočítávat hodnoty některých vlastností. Jak již bylo uvedeno v kapitole 2.2, MS Access používá ve svém uživatelském rozhraní centimetry jako jednotku míry. Pokud však přistupujeme k vlastnostem objektů pomocí kódu VBA (viz kapitola 6.1), pak je měrnou jednotkou *twip* (1 cm je roven 567 twip). Jelikož se ve Windows Forms aplikacích používají pixely, musí být tyto jednotky přepočítávány: 1 pixel je roven 15 twip.

Dalšími vlastnostmi, které je třeba přepočítat, jsou barvy – např. barva písma, pozadí apod. V exportovaných souborech je barva uložená jako číslo v dekadickém zápisu, ze kterého potřebujeme získat standardní formát RGB. Pokud uvažujeme pro jednotlivé RGB barvy hodnoty v rozsahu 0 až 255, pak je zmíněné desítkové číslo rovno: $r + (g * 256) + (b * 65536)$, kde r, g a b jsou proměnné představující jednotlivé barevné složky (červená, zelená, modrá). Pokud toto číslo převedeme do hexadecimální soustavy, rozsah hodnot bude 000000_{16} až $FFFFFF_{16}$, přičemž první dvě číslice s nejvyšší vahou reprezentují modrou barvu, prostřední zelenou a poslední dvě červenou. Pro tento převod lze použít bitových operací, které jsou použity ve výpisu kódu č. 10.

```
int red = decColor & 0xFF;
int green = (decColor >> 8) & 0xFF;
int blue = (decColor >> 16) & 0xFF;
```

Výpis 10: Převod barvy na RGB

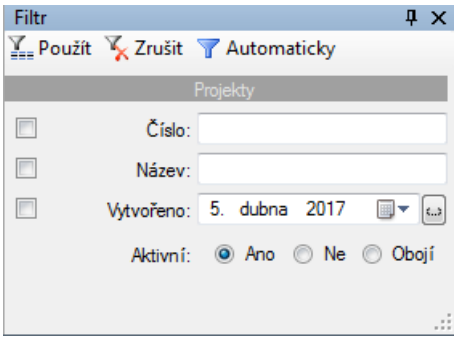
8.1.3.3 Převod formuláře na tabulkové zobrazení

Zdrojovým objektem z MS Access aplikace je datový list (viz kapitola 2.1.2), z něhož po převodu vznikne třída odvozená od `StandardGridControl` nebo `StandardEditableGridControl` (záleží, jestli je datový list editovatelný či nikoli). Ze zdrojového datového listu se převedou textová pole, zaškrtávací políčka a pole se seznamem na sloupce tabulky typu `DataGridViewTextBoxColumn`, `DataGridViewCheckBoxColumn` a `DataGridViewComboBoxColumn` komponenty `DataGridView`. Zmíněné ovládací prvky jsou v modelu AccessConverter reprezentovány třídami `TextBoxItem`, `CheckBoxItem` a `ComboBoxItem`. Všechny tyto třídy implementují rozhraní `ILabeledItem` (viz obrázek 10). V MS Access aplikacích mají tyto ovládací prvky přidružené popisky, které definují

názvy sloupců v tabulce. Taková vazba je v exportovaných souborech uložena jako vlastnost *Parent*, která obsahuje název nadřazeného prvku. U popisků může být v této vlastnosti uložen název formuláře, na kterém se nachází (pak popisek není vázaný k žádnému ovládacímu prvku), a nebo právě název přidruženého ovládacího prvku. Takové vazby jsou při převodu detekovány a ovládacím prvkům implementujícím rozhraní `ILabeledItem` se přiřazují popisky. Dále tyto prvky obsahují vlastnosti jako *ColumnWidth* a *ColumnOrder* pro nastavení šířky sloupce a jeho pořadí. Ukázka převedeného tabulkového zobrazení je vidět na obrázku 11.

Obrázek 11: Ukázka převedeného tabulkového zobrazení

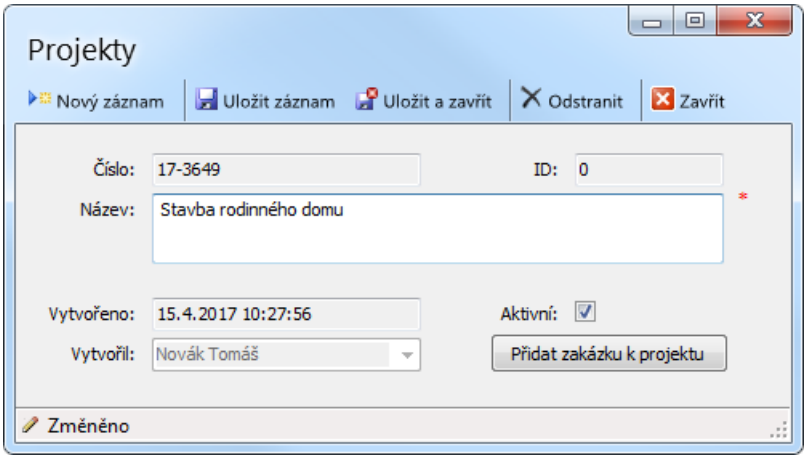
8.1.3.4 Převod formuláře na filtr Zdrojovým objektem je obvykle o rodičovský formulář příslušného datového listu. Z takového formuláře vznikne po konverzi třída dědící z `SqlFilterControl` (viz kapitola 7.1). Filtr musí být v převedené aplikaci vždy přidružen k jednomu tabulkovému zobrazení. Ze zdrojového formuláře budou veškerá textová pole, zaškrtávací políčka a pole se seznamem převedeny na filtry přidruženého tabulkového zobrazení. K těmto prvkům budou stejně jako u tabulkového zobrazení přiřazeny popisky jakožto viditelné názvy jednotlivých filtrů. Jelikož filtrování je v MS Access řešeno pomocí zdrojového kódu VBA, tato pole nebudou napojena na konkrétní atributy filtrované tabulky, bude tedy nutné je dodatečně napojit manuálně. Ukázka převedeného filtru je vidět na obrázku 12.



Obrázek 12: Ukázka převedeného filtru

8.1.3.5 Převod formuláře na detail

Detail je část modulu, jež je standardně zobrazena po dvojkliku na některý záznam v tabulkovém zobrazení. Hodnota primárního klíče označeného záznamu se předá jako spouštěcí parametr a otevře se detail záznamu. Takový dialog pak plní podobnou funkci jako MS Access formulář (viz kapitola 2.1.1), který má definovaný datový zdroj. Na převedeném formuláři budou napojeny všechny prvky obsažené na formuláři k tabulce v databázi a budou k dispozici funkce vložení, aktualizace a smazání záznamu, tedy standardní CRUD operace. Pokud takový formulář bude obsahovat vnořený datový list, pak bude tento datový list převeden na komponentu `DataGridView`, ale nebude napojen na žádný zdroj dat. Napojení bude muset provést dodatečně programátor, který bude převod provádět. Ukázka detailu je vidět na obrázku 13.



Obrázek 13: Ukázka převedeného detailu

V následující tabulce je uveden vždy jeden ovládací prvek Windows Forms (viz kapitola 3.1) a k němu přiřazen vzor MS Access aplikace (viz kapitola 2.2). Takové transformace se využívá pro převod uživatelského rozhraní detailu i obyčejného formuláře.

Windows Forms	MS Access	Komentář
`Label`	*Popisek*	
`LinkLabel`	*Popisek*	Pokud má popisek definovanou vlastnost *hypertextový odkaz*, pak se transformuje na `LinkLabel` a bude otevírat zadanou webovou stránku. Pokud se bude jednat o emailovou adresu, otevře výchozího poštovního klienta.
`TextBox`	*Textové pole*	
`Button`	*Příkazové tlačítko*	
`CheckBox`	*Zaškrtávací políčko*	
`RadioButton`	*Přepínač*	

Pokračování tabulky na další stránce

Windows Forms	MS Access	Komentář
`ComboBox`	*Pole se seznamem*	`ComboBox` bude i po převodu napojen na zdroj dat. Zobrazuje-li *pole se seznamem* více sloupců, `ComboBox` bude zobrazovat pouze první z nich.
`GroupBox`	*Obdélník*	Aby byly viditelné prvky, které *obdélník* původně orámovával, musí být `GroupBox` umístěn na pozadí formuláře („Send to Back“), jinak by prvky překrýval. V generovaném design kódu formuláře to znamená přidat `GroupBox` na formulář vždy jako poslední.
`TabControl`	*Karta*	
`DataGridView`	Datový list	Datový list není ovládací prvek, ale jedná se přímo o typ zobrazení formuláře. Pokud bude převáděný formulář obsahovat vnořený datový list, v nové aplikaci se vloží komponenta `DataGridView` s vlastnostmi vnořeného formuláře.

Tabulka 14: Převod ovládacích prvků na formuláři

MS Access formulář obsahuje části jako jsou záhlaví, tělo a zápatí. Pozice prvků na formuláři je definovaná v rámci části, na které je prvek umístěn. Nově vytvořená Windows Form aplikace takové části neobsahuje, a proto je nutné vždy určit, v jaké části se prvek nachází a přepočítat jeho pozici.

8.2 Převod pomocí nástroje Access Converter

Z hlediska uživatele bude při konverzi aplikace MS Access nutné pomocí průvodce provést následující kroky:

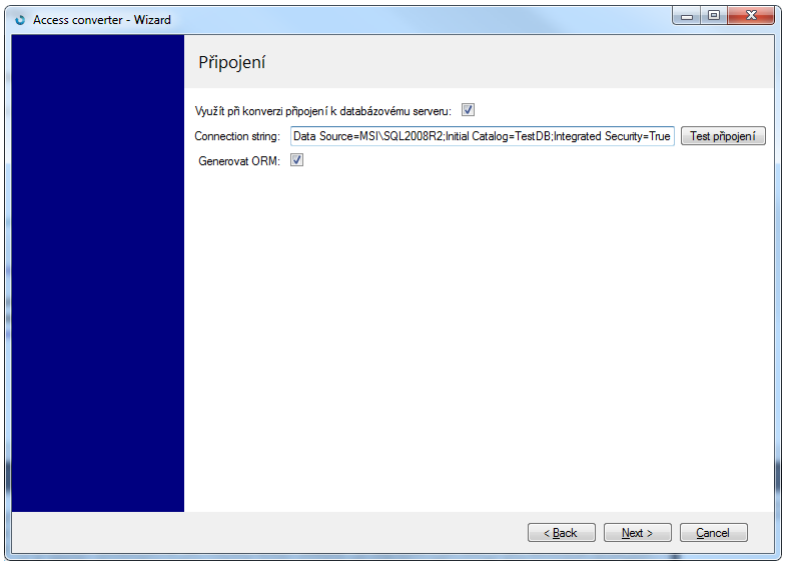
1. Spustit aplikaci MS Access určenou ke konverzi v editovatelném módu (do tohoto módu se lze standardně dostat podržením klávesy *SHIFT* při spouštění MS Access aplikace).
2. V sekci *Moduly* přidat nový modul a zkopírovat do něj obsah souboru `AccessModule.txt` (tento soubor je obsažen v příloze).
3. V nově vytvořeném modulu poté spustit metodu `Export`, načež se objeví dialogové okno pro zadání místa uložení exportovaných vlastností aplikace. Ve zvoleném umístění MS Access vytvoří adresář s názvem `AccessExportedData`, kde se uloží exportované vlastnosti aplikace. Tento export je popsán v kapitole 6.1.

4. Spustit aplikaci Access Converter a pomocí průvodce nastavením postupně zadávat požadované parametry potřebné k samotné konverzi. Parametry jsou následující:

   - Umístění adresáře `AccessExportedData`, které bylo zvoleno v bodě 3.
   - Cesta k adresáři cílové aplikace, do které se budou přidávat vygenerované soubory. V případě, že jde o generování nové aplikace lze využít šablonu s názvem „Template“, která je součástí příloh této práce. Tato šablona je prázdná Windows Forms aplikace, která má v referencích přidané knihovny SmartISLib a WeiFen Luo's DockPanelSuite. Šablona dále obsahuje ikony, na které se nová aplikace může odkazovat.
   - Dále je možné nastavit, zda se budou vygenerované soubory importovat do projektu Visual Studio. Pokud tato vlastnost není nastavena, soubory se pouze vygenerují, uloží do příslušného adresáře, ale v souboru `.csproj` na ně nebude naveden odkaz, a tudíž je bude nutné dodatečně přidat.

   Potvrzením tohoto kroku Access Converter ze souboru `ObjectDB.txt` získá připojovací řetězec k databázi, který je možno využít buďto již při konverzi aplikace (např. generování ORM) a nebo pokud bylo v předchozím bodě nastaveno „Generovat novou aplikaci - ano“, vloží se připojovací řetězec přímo do konfiguračního souboru cílové aplikace.

5. V aplikaci Access Converter zvolit, zda se má využívat připojení k databázi již při konverzi. V tomto kroku může uživatel upravit samotný připojovací řetězec, otestovat funkčnost připojení k MS SQL Serveru a zvolit, zda se budou do cílové aplikace generovat třídy ORM. Karta „Připojení“ je vidět na obrázku 14.



Obrázek 14: Nastavení připojení

Access Converter provede otestování připojení k databázi (pokud je připojení povoleno uživatelem) a v případě neúspěšného pokusu o připojení vypíše chybové hlášení a nedovolí uživateli pokračovat.

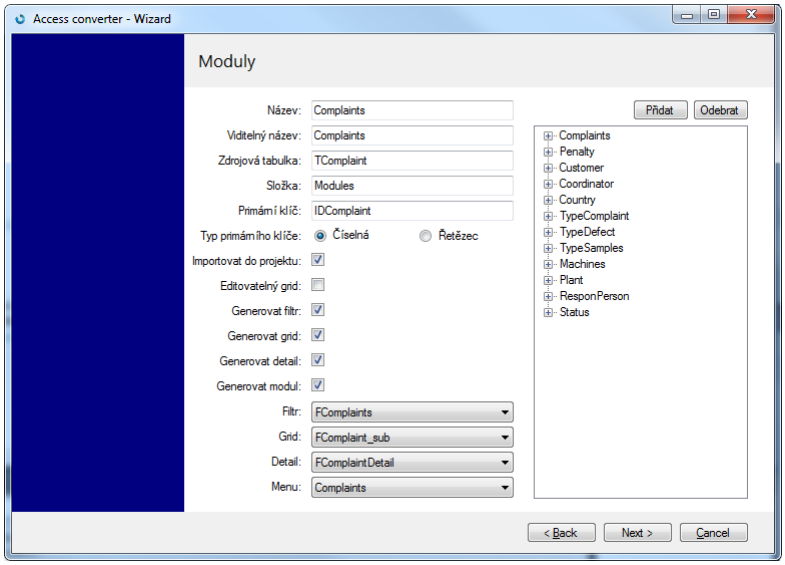
Z vygenerovaných souborů Access Converter zkompletuje objekty hlavní nabídky. Na základě těchto objektů se automaticky vytvoří moduly (viz kapitola 8.1.2).

6. Uživatel může dále uspořádat automaticky vytvořené moduly nebo vytvořit nové. Modulům je možné nastavit následující parametry:

   - *Název* bude použit pro pojmenování vygenerovaných tříd a vygenerovaných souborů.
   - *Viditelný název* bude zobrazován přímo uživateli, který bude s převedenou aplikací pracovat.
   - *Složka* – adresář, ve kterém bude modul uložen.
   - *Zdrojová tabulka*, *Primární klíč* a *Typ primárního klíče* jsou relevantní, pouze pokud modul obsahuje detail (viz kapitola 8.1.3.5). *Primární klíč* slouží pro specifikaci zvoleného záznamu v tabulkovém zobrazení a jeho hodnota se předá do formuláře s detailem. Zbylé dva parametry jsou nezbytné pro správnou funkci detailu.

     Pokud bylo povoleno připojení k databázi během konverze, pak se tyto parametry automaticky doplní pomocí analýzy dotazu, který byl použit pro naplnění tabulkového zobrazení daty.
   - Zaškrtávací tlačítka určují, zda se mají vytvářet jednotlivé části modulu. Tato možnost se využije zejména při konverzi do již existující aplikace. Tlačítko *Importovat do projektu* značí, zda bude odkaz zdrojových souborů modulu vkládat do projektu (souboru `.csproj`).
   - Pomocí rozbalovacích nabídek umístěných v dolní části karty lze přeorganizovávat moduly. Nabídka *Grid* obsahuje všechny datové listy. Nabídky *Filtr* a *Detail* čítají všechny ostatní formuláře. Jelikož detail nikdy není přiřazen programem automaticky, je zapotřebí zvolit příslušný formulář manuálně. V nabídce *Menu* jsou pak obsaženy jednotlivá menu tlačítka pro spouštění daného modulu.

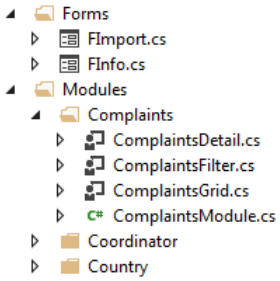
Na obrázku 15 je vidět karta průvodce pro nastavení tohoto kroku.



Obrázek 15: Nastavení modulů

7. Následuje nastavení formulářů, které zobrazuje všechny samostatné formuláře původní aplikace. Nastavení na této kartě je obdobné jako u modulů v předchozím bodě.

Tímto výčtem tedy končí práce uživatele a o všechno ostatní se postará aplikace Access Converter. Aplikace vygeneruje soubory modulů a formulářů. Podle zvoleného nastavení se vytvoří adresářová struktura se zdrojovými soubory. Příklad takové struktury je vidět na obrázku 16.



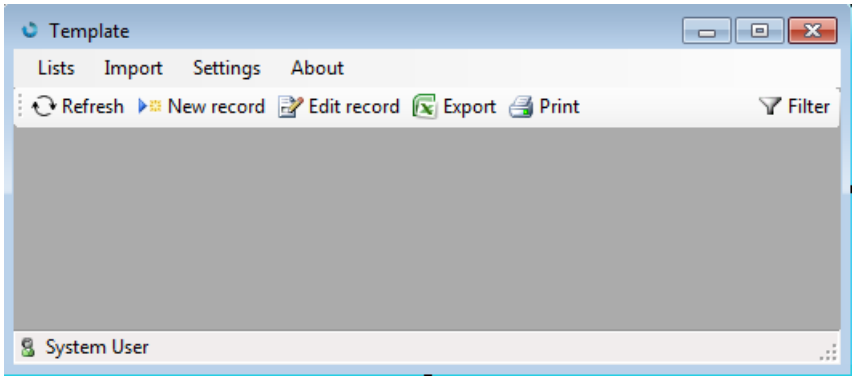
Obrázek 16: Ukázka adresářové struktury vygenerovaných modulů a formulářů

Pokud nebyla zvolena možnost generování nové aplikace, proces převodu zde končí. V opačném případě se generují třídy nezbytné pro chod aplikace:

- `ExeSettings` - Zajišťuje načítání připojovacího řetězce (*connection string*) z konfiguračního souboru.
- `ExceptionHandler` - Zajišťuje výpis chybových hlášeních v aplikaci. Úpravou této třídy lze zpracovávat chybová hlášení podle potřeby.

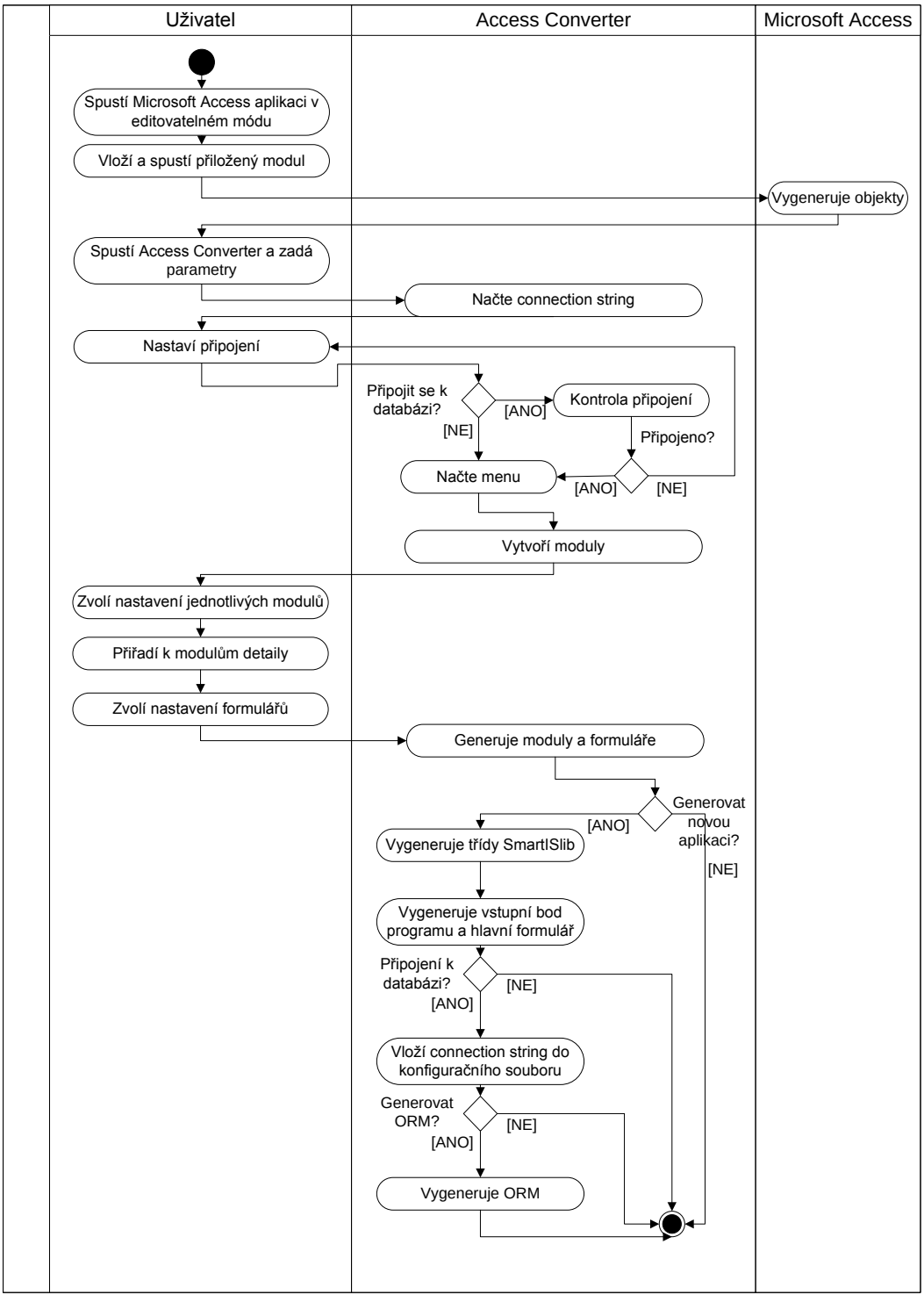
- `UserSettingsManager` - Lze doprogramovat ukládání a načítání uživatelského nastavení převedené aplikace.
- `User` - Složí pro autentifikaci uživatele. Ve výchozí implementaci pouze získá výsledek dotazu `SELECT SYSTEM_USER` zaslaný na databázi při spuštění aplikace.

Následuje generování souboru `Program.cs`, který bude obsahovat vstupní bod programu a generování hlavního formuláře `FormMain`. Tento formulář se objeví při startu aplikace a bude obsahovat rozhraní pro spouštění jednotlivých modulů. Na obrázku 17 je vidět příklad takového formuláře. Pokud bylo uživatelem povoleno připojení k databázi, Access Converter vloží připojovací řetězec do konfiguračního souboru. Za předpokladu, že byla zatržena volba generování ORM, Access Converter vygeneruje soubory `CachedFunctions.cs`, `Function.cs`, `Procedures.cs` a `Tables.cs` do adresáře `Data`.



Obrázek 17: Ukázka hlavního formuláře

Celý proces převodu shrnuje diagram na obrázku 18.



Obrázek 18: Diagram znázorňující postup konverze

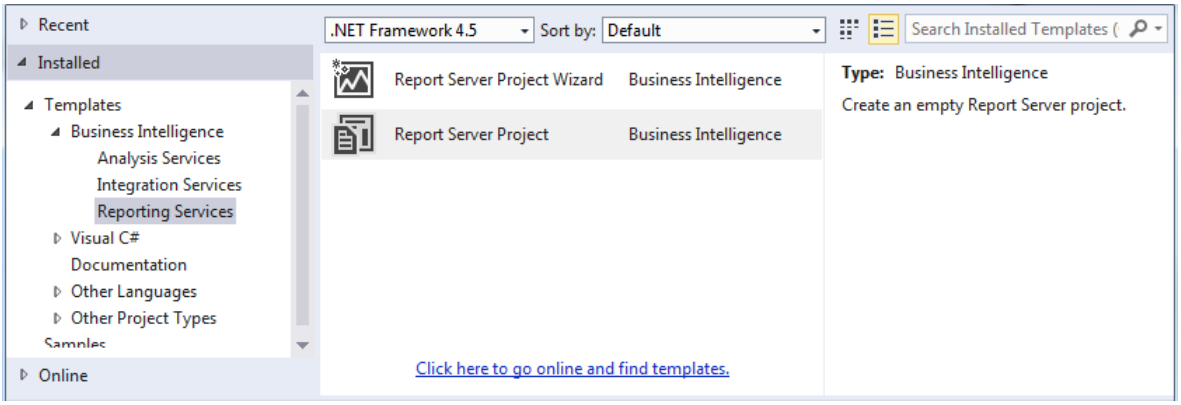
9 Převod tiskových sestav

Tiskové sestavy v MS Access představují jeden z výstupů aplikace. V aplikačním rámci .NET Framework lze vytvářet takovéto tiskové sestavy pomocí reportovacího nástroje SQL Server Reporting Services. Jeho samostatná komponenta Microsoft Report Viewer zajišťuje vytváření tiskových sestav, které lze buď tisknout, zobrazovat do náhledu nebo exportovat do formátů DOC, XLS a PDF. Pro převod tiskových sestav z MS Access do reportů .NET je tento nástroj ideální a dokonce lze použít již existující řešení od firmy Microsoft. Takové řešení využívá službu SQL Server Reporting Services (SSRS) jako hostující aplikaci tiskových sestav. Tato služba je k dispozici ve formě doplňku pro SQL Server a od verze SQL Server 2005 je spolu s Analysis Services a Integration Services součástí platformy Business Intelligence Development Studio [12]. Tisková sestava bude tedy spouštěna přímo na databázovém serveru a je tedy možno ji využít jak v desktopových, tak i ve webových aplikacích. Pro tuto konverzi je nutné mít nainstalováno:

- MS Access
- Microsoft Visual Studio včetně modulu Business Intelligence
- Přístup k MS SQL serveru se službou SQL Server Reporting Services

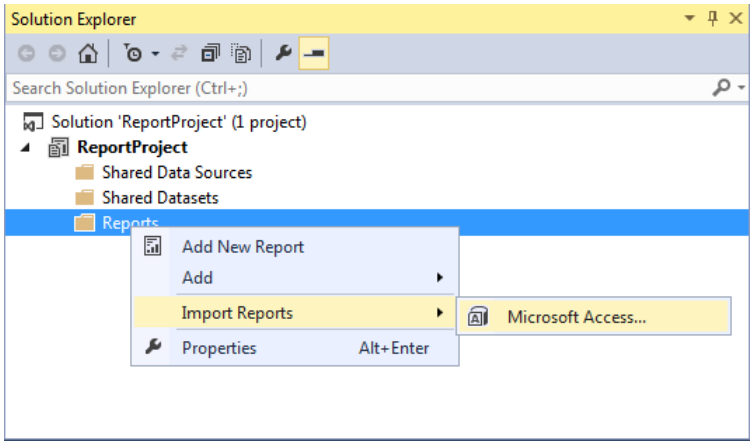
9.1 Postup konverze

1. V aplikaci Microsoft Visual Studio vytvoříme reportovací projekt (viz obrázek 19).



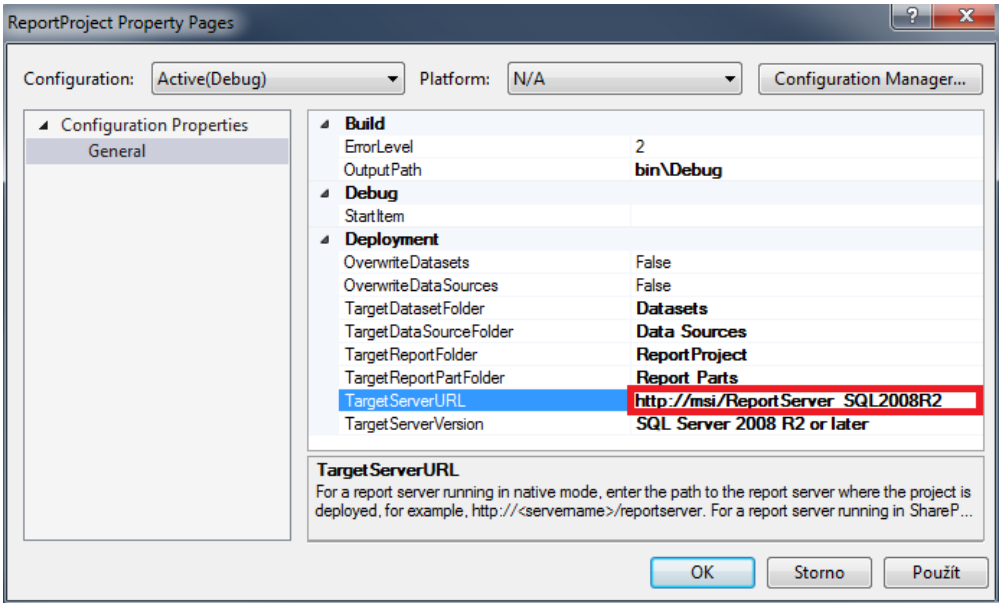
Obrázek 19: Vytvoření reportovacího projektu

2. Na panelu „Solution exlorer“ přes kontextové menu položky *Reports* zvolíme *Import Reports / Microsoft Access* (viz obrázek 20).
3. V zobrazeném dialogu zvolíme soubor typu MDB nebo ADP (aplikace MS Access), která obsahuje tiskové sestavy pro konverzi.



Obrázek 20: Import tiskových sestav z aplikace MS Access

4. Na panelu „Solution exlorer“ klikneme pravým tlačítkem na projekt, zvolíme možnost *Properties* a zadáme URL adresu reportovací služby, kterou chceme využívat pro spouštění tiskových sestav (jak získat tuto adresu URL je popsáno v kapitole 9.2). Na obrázku 21 jsou vidět vlastnosti reportovacího projektu s vyznačenou URL adresou.



Obrázek 21: Vlastnosti reportovacího projektu

5. Nakonec v hlavním menu klikneme na *Build*, zvolíme možnost *Deploy solution* a tiskové sestavy se nahrají na SQL server.

Nyní jsou všechny tiskové sestavy převedeny z MS Access aplikace do speciálního definičního jazyka RDL (Report Definition Language - jazyk pro návrh reportů). [11] Report, respektive soubor s kódem jeho návrhu, obsahuje tři druhy informací:

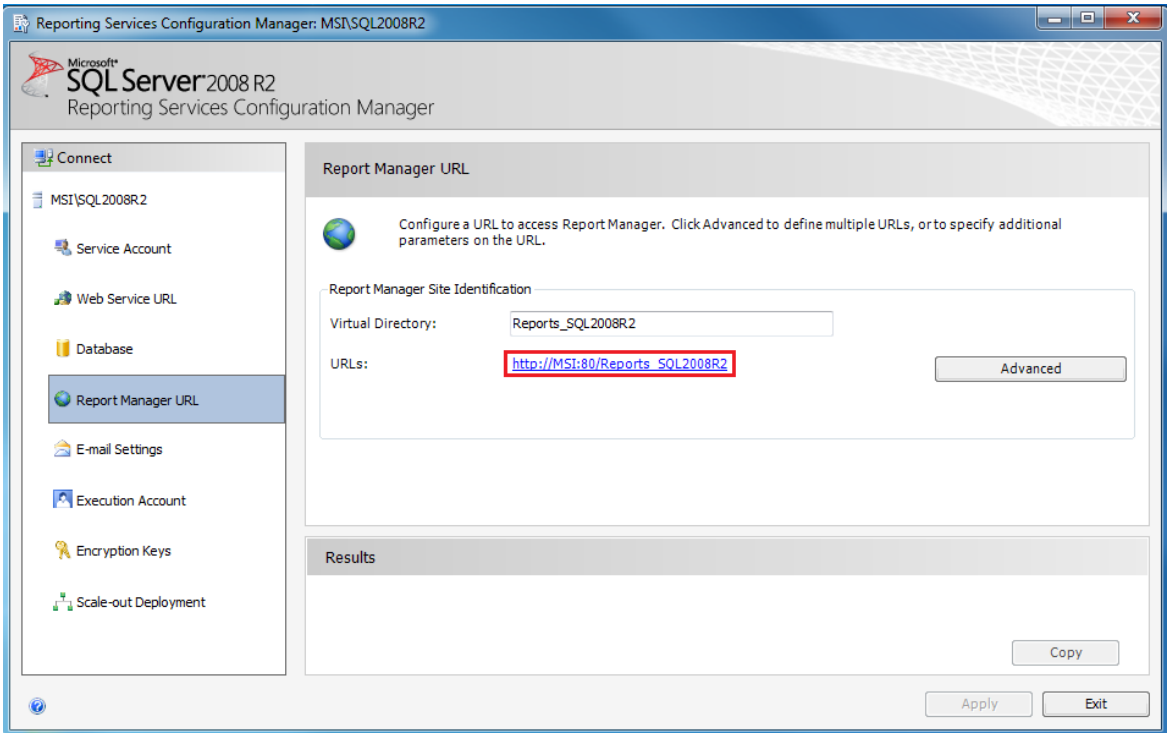
- Data, případně informace, jak se k požadovaným datům připojit (v tomto případě SQL dotaz), struktury dat.
- Návrhové schéma, které obsahuje informace o tom, v jaké podobě se data budou prezentovat.
- Vlastnosti tiskových sestav a jeho jednotlivých prvků.

9.2 SQL Server Reporting Services (SSRS)

Reportovací služby jsou určitou formou bezstavového serveru, jenž je součástí Microsoft SQL serveru. Je implementovaný jako služba operačního systému Microsoft Windows. Tento server spravuje metadata (definiční data o datech), definice objektů, způsob jejich zobrazení a podobně. Tato data mají reportovací služby uložené ve vlastní databázi, která je také ve správě Microsoft SQL serveru. [12]

Report server je možné ovládat přes web - nachází se ve virtuálním adresáři a je přístupný přes URL adresu (viz obrázek 22).

Nástroj pro konfiguraci reportovacích služeb je přístupný přes menu operačního systému: *Start -> Programs -> Microsoft SQL server -> Configuration Tools -> Reporting Services Configuration Manager*. Prostřednictvím tohoto vizuálního nástroje lze získat, v případě potřeby i změnit, většinu důležitých parametrů pro práci reportovacích služeb. Pro použití reportů je nutné získat URL adresu reportovací služby, která se nachází v záložce *Report Manager URL*.

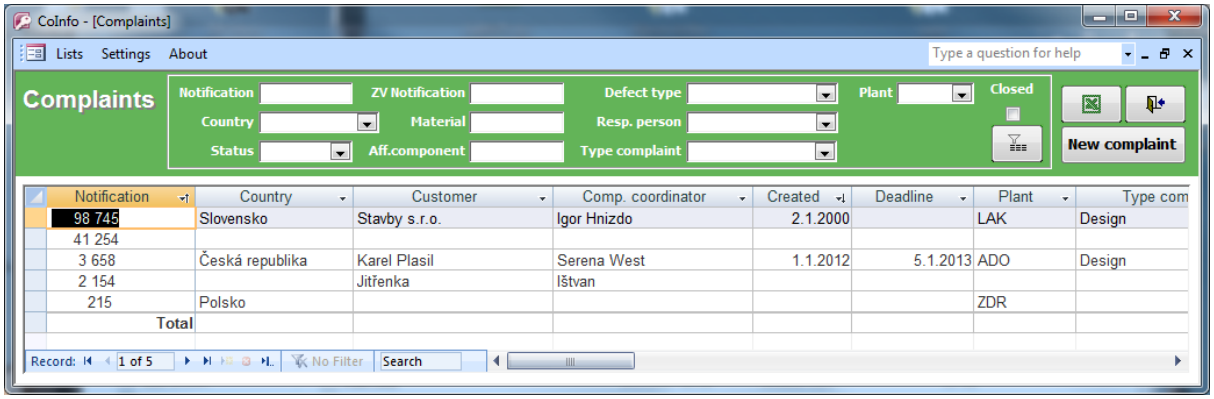


Obrázek 22: URL adresa reportovací služby

10 Převod vzorové MS Access aplikace

V příloze obsažená vzorová aplikace s názvem CoInfo eviduje seznam reklamací, které byly zaslány do výroby. Tato aplikace komunikuje s MS SQL Serverem a databází nazvanou TestDB, jejíž záloha je obsažena v příloze pod názvem „TestDB.bak“ (verze MS SQL Server 2008 R2). V databázi jsou obsažena i testovací data.

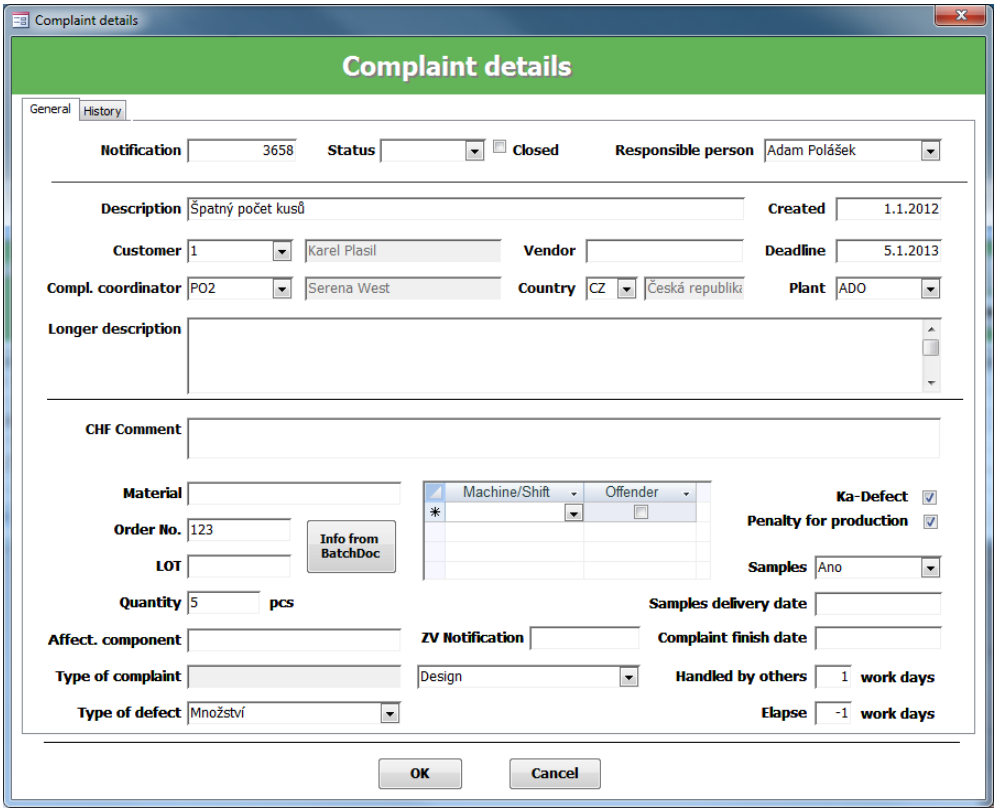
Aplikace obsahuje standardní MS Access menu, která čítá tyto nabídky: *Lists*, *Settings* a *About*. V nabídce *Lists* se nachází položka *Complaints*, spouštějící stejnojmenný formulář, který obsahuje tabulku s reklamacemi. Součástí tohoto formuláře jsou i ovládací prvky, které slouží jako filtrovací pole pro tabulku. Tento formulář je vidět na obrázku 23.



Obrázek 23: Vzorová MS Access aplikace – formulář Complaints

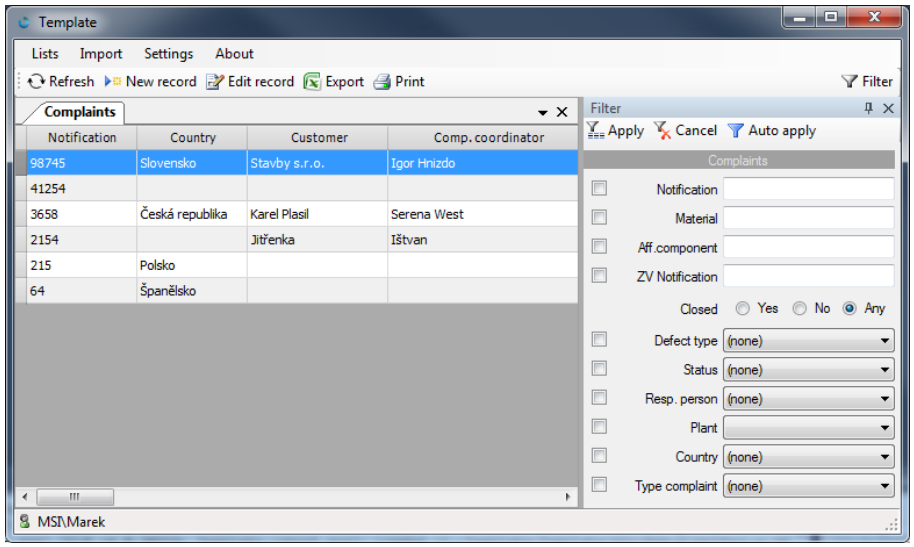
Při dvojkliku na řádek v tabulce se otevře detail záznamu, který je vidět na obrázku 24. Na detailu jsou zobrazeny dvě překrývající se karty *General* a *History*. Na kartě *General* lze upravovat otevřený záznam nebo přiřazovat k reklamaci směnu odpovědnou za definovanou vadu. Karta *History* obsahuje tabulku, kde je uveden seznam změn daného záznamu.

Nabídka *Lists* obsahuje ještě položku *Penalty*, která otevírá seznam prohřešků, kterých se dopustila konkrétní směna. I tento přehled nabízí možnost filtrování položek, ale neobsahuje žádný detail ani možnost editace záznamů. Další nabídkou v hlavním menu je *Settings*, obsahující deset položek odkazujících na editaci číselníků v databázi prostřednictvím editovatelných datových listů (viz kapitola 2.1.2). Poslední položka v menu je *About*, která spouští informativní formulář o verzi aplikace.

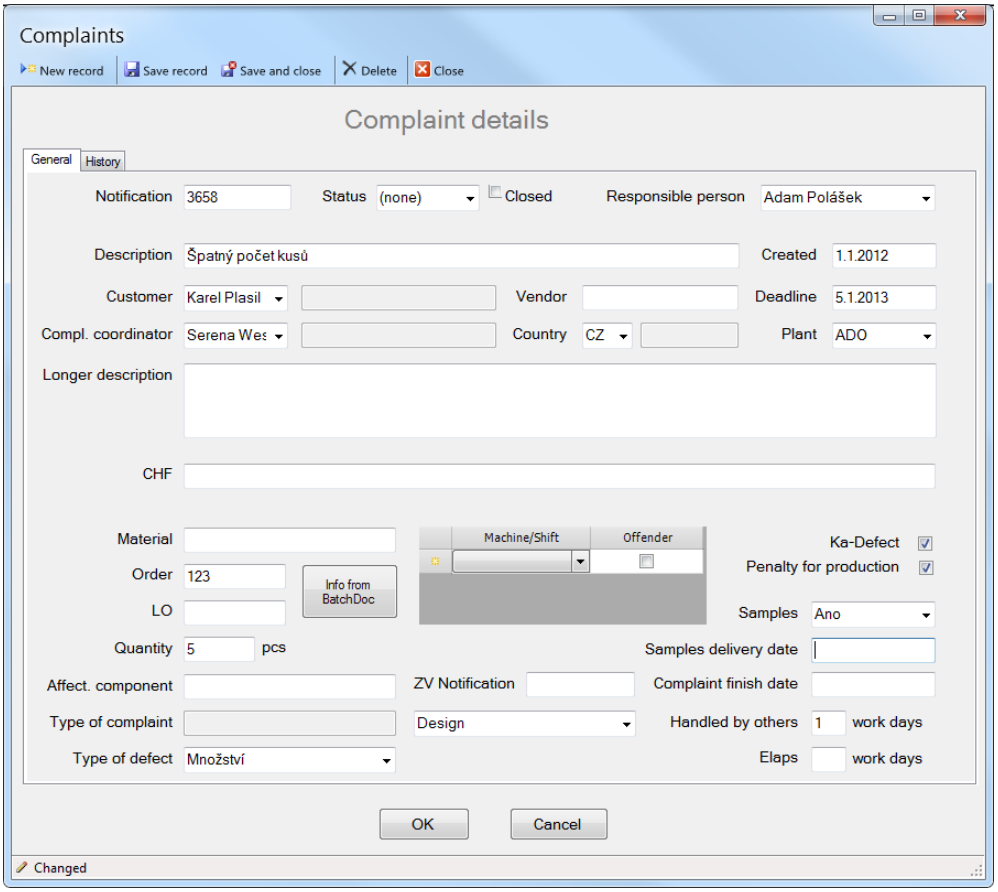


Obrázek 24: Vzorová MS Access aplikace - detail Complaints

Při převodu této aplikace je zapotřebí dodržet postup konverze popsaný v kapitole 9.1 a v bodě 6 přiřadit modulu *Complaints* detail, který se jmenuje *FComplaintDetail*. Převedená aplikace s otevřeným modulem *Complaints* je vidět na obrázku 25 a detail tohoto modulu na obrázku 26.



Obrázek 25: Převedená aplikace - formulář Complaints



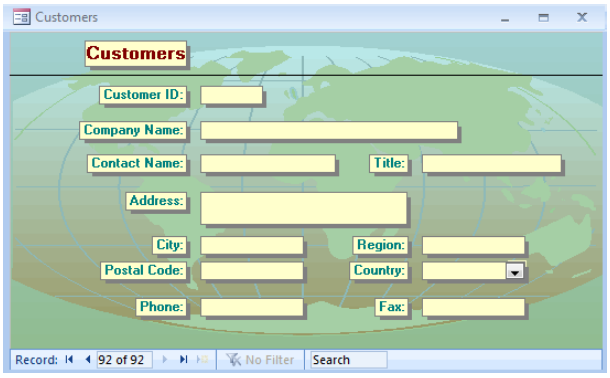
Obrázek 26: Převedená aplikace - detail Complaints

10.1 Porovnání vytvořené aplikace s existujícími nástroji

Testované konverzní nástroje Access Whiz (viz kapitola 4.1) a Evolution MS Access to VB.NET Converter (viz kapitola 4.2) se snaží o co nejvěrnější převod MS Access aplikací do prostředí .NET. Nicméně takový převod je velice náročný. Narážíme tady na řadu problémů, jako je jiný programovací jazyk, odlišná komunikace se SŘBD a celkově odlišný přístup při vývoji aplikací. Zmíněné konverzní nástroje zdaleka nedosahují dokonalosti a před reálným nasazením převedených aplikací bude ve velké míře nutné převedené aplikace upravit.

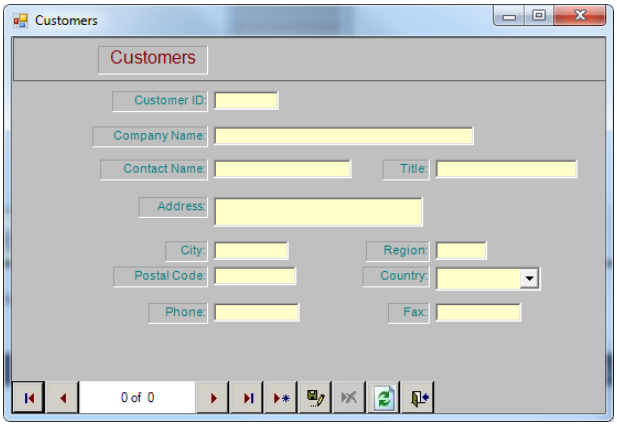
Vytvořený nástroj Access Converter spíše slouží pro zautomatizování procesů při vývoji nové aplikace, která je postavena na staré MS Access aplikaci. Uživatel – programátor, který bude převod provádět, bude muset sestavovat moduly z původních formulářů a pokročilou funkčnost na nich obsaženou manuálně doprogramovat.

Na obrázku 27 je vidět formulář Customers z MS Access aplikace Northwind, který byl převeden pomocí všech zmíněných konverzních nástrojů.



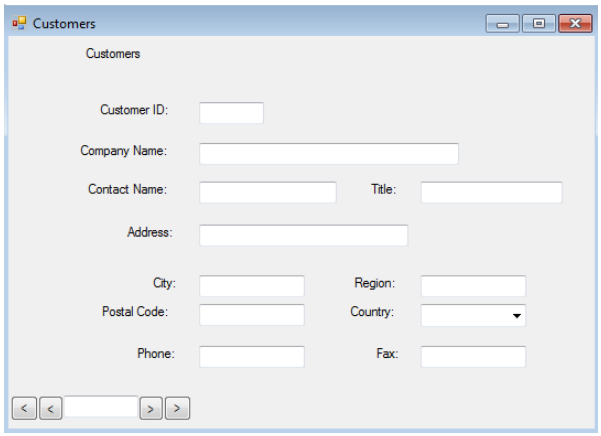
Obrázek 27: Původní formulář v aplikaci MS Access

Na převedeném formuláři pomocí Access Whiz (viz obrázek 28) je vidět, že tento nástroj převedl všechny prvky obsažené na formuláři i s jejich barevným provedením. Na spodní části formuláře jsou převedená funkční navigační tlačítka.



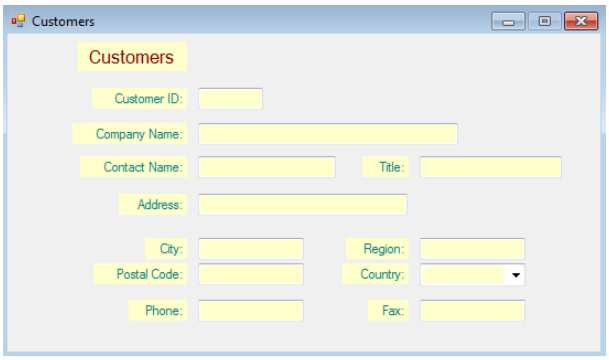
Obrázek 28: Formulář převedený pomocí nástroje Access Whiz

Formulář převedený pomocí nástroje Evolution MS Access to VB.NET Converter také převedl všechny ovládací prvky ovšem bez jejich barevné reprezentace (viz obrázek 29). Ve spodní části formuláře jsou obsažena navigační tlačítka, která mají definované tělo metody `OnClick`, ale nutno dodat, že aplikace nebyla převedena do spustitelného kódu a nebylo možné tyto funkce otestovat.



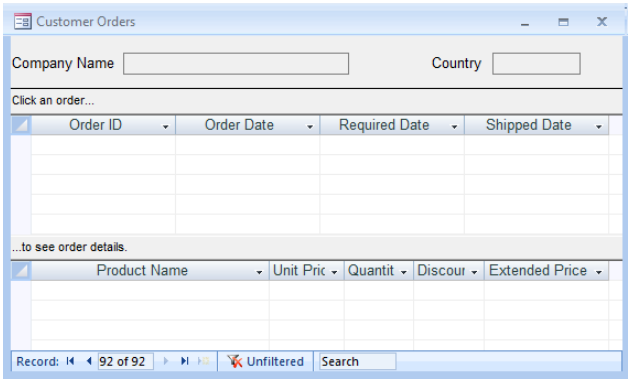
Obrázek 29: Formulář převedený pomocí nástroje Evolution MS Access to VB.NET converter

Vytvořený nástroj Access Converter také převedl všechny ovládací prvky na formuláři i s jejich barevnou reprezentací. Jelikož vzorovou aplikací pro tento formulář byla aplikace Northwind, která využívá databázi Microsoft Jet (jedná se o aplikaci typu MDB, pro kterou Access Converter nebyl navrhnut), CRUD operace na převedeném formuláři nejsou k dispozici. Převedený formulář je vidět na obrázku 30.

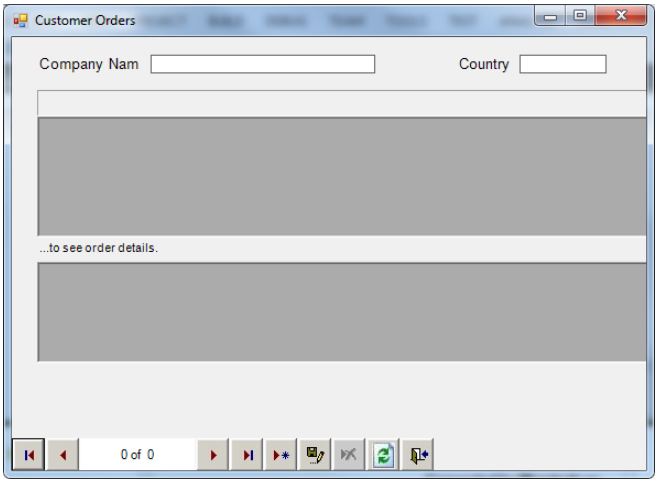


Obrázek 30: Formulář převedený pomocí vytvořeného nástroje Access Converter

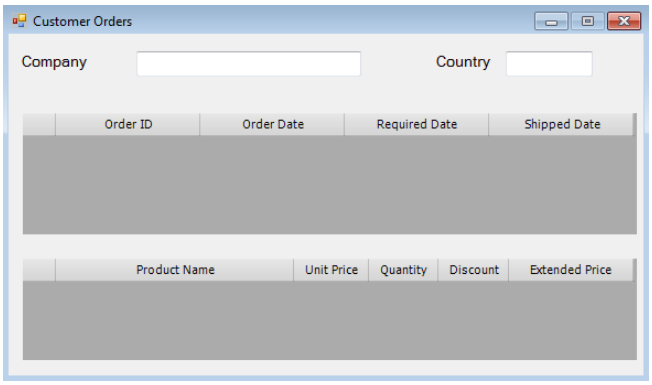
Na obrázku 31 je vidět formulář Customer Orders se dvěma vnořenými datovými listy. Jelikož pomocí nástroje Evolution MS Access to VB.NET Converter se nepodařilo převést ani jeden formulář s vnořeným datovým listem, jsou na následujících obrázcích vidět pouze formuláře převedené pomocí Access Whiz (obrázek 32) a Access Converter (obrázek 33).



Obrázek 31: Původní formulář s vnořenými datovými listy v aplikaci MS Access



Obrázek 32: Formulář s vnořenými datovými listy převedený pomocí nástroje Access Whiz



Obrázek 33: Formulář s vnořenými datovými listy převedený pomocí vytvořeného nástroje Access Converter

11 Závěr

První část práce byla věnována rozboru objektů v aplikacích MS Access, jejich komunikaci se SŘBD a možnostem, jak programově získat informace o těchto objektech. Dále zde byly popsány ovládací prvky Windows Forms, které jsou využívány v převedených aplikacích.

Pro analýzu MS Access aplikací určených k převodu byl vytvořen modul v jazyce VBA, který exportuje strukturu aplikací do externích souborů. Tyto soubory jsou pak vstupem vytvořeného nástroje pro konverzi MS Access aplikací. Převedené aplikace pomocí tohoto nástroje jsou generované v jazyce C# a využívají kromě standardních knihoven .NET i externí knihovny popsané v této práci. Vytvořený konverzní nástroj není určen pro převod tiskových sestav. Pro takovou konverzi lze využít již existující řešení od společnosti Microsoft, které bylo v této práci rozebráno, a byl zde uveden i postup, jak této konverze dosáhnout.

Vytvořený konverzní nástroj využívá inovativního přístupu k převodu MS Access aplikací, spočívajícího v detekci ucelených částí původní aplikace a jejich převodu do definovaných modulů. Detekce těchto částí se provádí poloautomaticky, tzn. konverzní nástroj navrhne možné uspořádání modulů, a je na uživateli, jestli navržené řešení využije nebo vytvoří uspořádání jiné. Předpokladem pro úspěšný převod je například uživatelova znalost původní MS Access aplikace v kombinaci se správným nastavením parametrů při provádění konverze.

Nicméně plně automatického převodu MS Access aplikací do prostředí .NET nebylo dosaženo a programátor musí po převodu část funkcionality doprogramovat manuálně. Totéž platí i pro existující konverzní nástroje, které byly otestovány.

Za cenu neúplné konverze získáme výsledné aplikace v prostředí .NET, které jsou snadno rozšiřitelné a pochopitelné pro další údržbu. Převedené aplikace mohou sloužit jako základní kostra pro možnost přidávání nových modulů a funkcí za využití aplikačního rámce .NET. Co dříve nebylo možné v původních aplikacích MS Access provést, lze nyní implementovat.

Literatura

[1] BELKO, Peter. Microsoft Access 2013: podrobná uživatelská příručka. Brno: Computer Press, 2014. ISBN 978-80-251-4125-0.

[2] VIESCAS, John. Mistrovství v Microsoft Access 2000: kompletní průvodce efektivního uživatele i tvůrce databází. Praha: Computer Press, 2000. Databáze. ISBN 80-7226-274-2.

[3] SQL: Access to SQL Server [online]. Berkeley, CA: Apress, 2002 [cit. 2017-03-20]. ISBN 978-1-893115-30-9.

[4] JENNINGS, Roger. Special edition using Microsoft Access 2002. Indianapolis, IN: Que Publishing, 2002. ISBN 978-0-7897-2510-3.

[5] PETZOLD, Charles. Programování Microsoft Windows Forms v jazyce C#: [vytváříme uživatelské rozhraní aplikací]. Brno: Computer Press, 2006. ISBN 80-251-1058-3.

[6] POWELL, Robert a Richard WEEKS. C# and the .NET Framework: the C++ perspective. Indianapolis, Ind.: SAMS, c2002. ISBN 0-672-32153-x.

[7] BROWN, Erik E. Windows Forms in action. 2nd ed. Greenwich, Conn.: Manning, c2006. ISBN 1932394-65-6.

[8] JENNINGS, Roger. Professional ADO.NET 3.5 with LINQ and the Entity Framework. Indianapolis, IN: Wiley Pub., c2009. Wrox professional guides. ISBN 978-0-470-18261-1.

[9] FOWLER, Martin. Patterns of enterprise application architecture. Boston: Addison-Wesley, c2003. ISBN 0-321-12742-0.

[10] COUCH, Andrew. Microsoft Access 2010: VBA programming inside out. Sebastopol, Calif.: O'Reilly Media, c2011. Inside out (Redmond, Wash.). ISBN 978-0-7356-5987-2.

[11] SAYED, Asif. Client-side reporting with Visual Studio in C#. New York: Distributed to the Book trade worldwide by Springer-Verlag New York, c2007. ISBN 978-1-59059-854-2.

[12] LACKO, Ľuboslav. Business Intelligence v SQL Serveru 2005: reportovací, analytické a další datové služby. Brno: Computer Press, 2006. ISBN 80-251-1110-5.

A Obsah CD

Přiložené CD obsahuje tyto adresáře:

1. Aplikace Access Converter
2. Exportované objekty z aplikace CoInfo
3. MS Access modul pro exportování objektů
4. Převedená aplikace CoInfo pomocí nástroje Access Converter
5. Template - šablona .NET aplikace
6. Testovací databáze k aplikaci CoInfo
7. Vzorová MS Access aplikace CoInfo
8. Zdrojový kód aplikace Access Converter